Spring 2018

Near-optimal Control of Switched Systems with Continuous-time Dynamics using Approximate Dynamic Programming

Tohid Sardarmehni
Southern Methodist University, tsardarmehni@smu.edu

NEAR-OPTIMAL CONTROL OF SWITCHED SYSTEMS WITH
CONTINUOUS-TIME DYNAMICS USING
APPROXIMATE DYNAMIC PROGRAMMING

Approved by:

Dr. Ali Heydari
Assistant Prof. of Mechanical Engineering

Dr. Yildirim Hurmuzlu
Prof. of Mechanical Engineering

Dr. Edmond Richer
Associate Prof. of Mechanical Engineering

Dr. Khaled Abdelghany
Associate Prof. of Civil and Environmental Engineering

Dr. Mohammad Khodayar
Assistant Prof. of Electrical Engineering

NEAR-OPTIMAL CONTROL OF SWITCHED SYSTEMS WITH CONTINUOUS-TIME DYNAMICS USING APPROXIMATE DYNAMIC PROGRAMMING

A Dissertation Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Mechanical Engineering

by

Tohid Sardarmehni

B.Sc., Mechanical Engineering, Shahid Bahonar University of Kerman
M.Sc., Mechanical Engineering, Tabriz University

May 19, 2018

ACKNOWLEDGMENTS

As the author of this study, I am always grateful for the support and insight I received from my supervisor Dr. Ali Heydari. Also, I appreciate the support of my PhD advisory committee. This material is partially supported by the National Science Foundation under Grant 1509778 for which I am grateful. Also, I appreciate the financial support I received from the Department of Mechanical Engineering at the Southern Methodist University and the South Dakota School of Mines and Technology.

Sardarmehni, Tohid

B.Sc., Mechanical Engineering, Shahid Bahonar University of Kerman
M.Sc., Mechanical Engineering, Tabriz University

Near-optimal Control of Switched Systems with Continuous-time Dynamics Using Approximate Dynamic Programming

Advisor: Dr. Ali Heydari

Doctor of Philosophy degree conferred May 19, 2018

Dissertation completed May 19, 2018

Optimal control is a control method which provides inputs that minimize a performance index subject to state or input constraints [58]. The existing solutions for finding the exact optimal control solution such as Pontryagin's minimum principle and dynamic programming suffer from *curse of dimensionality* in high order dynamical systems. One remedy for this problem is finding near optimal solution instead of the exact optimal solution to avoid curse of dimensionality [31]. A method for finding the approximate optimal solution is through Approximate Dynamic Programming (ADP)[1] methods which are discussed in the subsequent chapters.

In this dissertation, optimal switching in switched systems with autonomous subsystems is studied. In order to derive the optimal solution, ADP method is used. Two iterative schemes, namely policy iteration and value iteration, from ADP methods are studied.

For policy iteration, continuous-time dynamics is considered and two different methods for solving the underlying Lyapunov equation as gradient descent and recursive least squares are studied. Also, a new method is introduced which tries to reduce the computational burden in policy iteration. For all the policy iteration based solutions, convergence of the iterations to the optimal solution and stability of the system during the training is studied. Additionally, three methods for implementing the policy iteration based solutions as offline training, online training, and concurrent training methods are discussed in details.

[1]The method is also called as Adaptive Dynamic Programming and Neuro Dynamic Programming [8].

For value iteration, the problem of deriving the optimal switching policy in anti-lock brake system of ground vehicles is investigated. For this purpose, a typical hydraulic brake system is used which can increase, decrease or hold the hydraulic braking pressure. The control goal is switching such that a variable in the model, called slip ratio, is regulated at its optimal value which leads to minimum stopping distance. For this problem, discrete time dynamics is used.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my family for their love and support.

Chapter 1

Introduction

Optimal control is a popular control method. In general, optimal control provides a control policy which minimizes a performance index subject to input or state constraints. The choice of the performance index is a designer choice. In general, optimal control tends to derive the control solution which globally minimizes the performance index. Also, in this study, the optimal policy is valid for all the initial conditions in a domain of interest and only feedback solutions are of our interest.

From the mathematical point of view, solutions to the underlying Hamilton-Jacobi-Bellman (HJB) equation provided the necessary and sufficient condition for optimality[1]. Hence, solving the optimal control problem is in fact solving the underlying HJB equation. However, solving the HJB equation analytically is very difficult and in many cases almost impossible.

A solution for optimal control problem is presented by minimum principle of Pontryagin and calculus of variations. However, this methods gets very complicated as the order of the system increases and it is limited to simplistic dynamics. A solution for optimal control problem is presented by minimum principle of Pontryagin and calculus of variations. However, this methods gets very complicated as the order of the system increases and it is limited to simplistic dynamics. In the presence of nonlinearities in the dynamics of the system, it is very difficult to solve the optimal control problem through minimum principle of Pontryagin. In order to deal with nonlinearities in the system, a closed form feedback solution for optimal control problems is suggested by Bellman which is called Dynamic Programming (DP). In summary, DP quantizes the state and control domain and generates a table of all possible solutions backward-in-time and save these solutions in the memory. In generating such table, DP uses the Bellman principle of optimality and saves

[1]This holds only in lumped systems. In case of distributed parameter systems, some more conditions should be satisfied in order to define optimality in the system.

the optimal cost-to-go for each point. For online control, DP refers to the generated table at each instant and chooses the optimal decisions.

In the first glance, it seems that DP can solve any optimal control problem. However, as the order of the system and the number of quantization grids increase, rapid access to memory becomes prohibitive. This problem is defined by Bellman as the *curse of dimensionality* in DP [58]. Also, DP is by nature offline and it solves backward-in-time [62]. In order to present a solution which solves online, forward-in-time and also does not suffer from the curse of dimensionality, Approximate Dynamic Programming (ADP) was introduced. In general, ADP methods use function approximators to approximate the optimal cost-to-go, namely critic, and sometimes optimal policy, namely actor. Then ADP methods use Reinforcement Learning (RL) methods to tune the unknown parameters of the function approximators.

Learning mechanism in cognitive agents is based on the interactions with their environment and the responses they receive from these interactions [62]. The agents repeat the actions which received awards and avoid the actions which received punishments. This *good-boy, bad-boy* strategy is called RL [103]. The ADP methods use iterative schemes for tuning the parameters of the function approximators and they use RL for this purpose, i.e., the ADP methods remembers the decisions which leads to lower costs (rewards) during the tuning and tends to repeat them in future.

The idea of using function approximators to approximate the optimal cost-to-go in ADP methods was first presented by Werbos [118] where linear in parameter neural networks were suggested to approximate the cost-to-go. The researchers after Werbos tried to classify these iterative methods more precisely. In summary, Value Iteration (VI) and Policy Iteration (PI) are two well-known and powerful iterative schemes that are widely used in ADP. Both VI and PI algorithms have two stages, namely policy evaluation and policy update. VI algorithm uses a simple recursion for finding the next value functions at each iteration in value evaluation. On the other hand, PI algorithm solves a Lyapunov equation at each step which casts more computational load per iteration. This feature of the PI algorithm is referred to as *full back-up* requirement of this algorithm which is a distinction from the *partial back-up* of the VI algorithm. Moreover, since PI algorithm solves a Lyapunov equation the evolving polices generated by this algorithm are stabilizing. This is not the case in

VI algorithm in general. The conditions for stabilizing feature of VI algorithm is recently studied in [37, 38].

From one point of view, one can categorize the reported ADP methods as Discrete-time (DT) solutions and Continuous-time (CT) solutions. Since the general scheme of this study is providing a CT solution with PI algorithm, a brief literature review is presented. In the first research works, offline training in PI algorithm with Galerkin method was presented in [6] and offline solution with least squares method was presented in [2]. For online training, a partially model free training method based on integral reinforcement learning was presented in [114] for linear systems and in [67, 111, 113] for nonlinear systems. Also the idea of simultaneously updating the actor and critic was introduced in [110] where full knowledge of the system dynamics was required and a gradient descent training law was used. In [9], the same idea as synchronous update of critic and actor was used with recursive least squares and the dynamics of the system were identified on the fly. In [20], a single online approximator was used which again used a gradient descent training law. All these contributions used Persistency of Excitation (PE) condition, borrowed from adaptive control techniques, to prove the uniformly ultimately boundedness of the critic weights error signal. However, PE condition is a restrictive assumption which cannot be evaluated online. Hence, the idea of using carefully selected offline data along with online data was introduced in [76] where the data that system already experienced is used again in training. The authors refer to this method as experience replay method. Also, in [54] similar idea is used where the offline data is not from the data system experienced before and the authors used the idea of simulation of an experience instead of experiencing a data to choose the offline data.

A major contribution of the present study is investigation of optimal control methods for control of switched systems. In general, hybrid systems are dynamical systems with both CT subsystems and DT events [136]. Switched systems are special class of hybrid systems in which the DT events define the switching instants and appropriate active subsystem, such that at each time instant only one subsystem is active [125, 126]. This definition of switched systems covers a broad range of problems in various engineering fields [30, 43, 49, 92]. In this study the subsystems are assumed to be autonomous which means there is no continuous control input in the subsystems.

Optimal control of switched systems is a challenging task due to discontinuous nature of the problem and lack of smooth control signals [39]. For optimal control of the switched systems with ADP, VI algorithm is used in [39] for regulation, in [36] for reference tracking, and in [91] for optimal control of switched systems with homogenous subsystems. Meanwhile, PI algorithm is used in [69] for optimal control of switched systems with controlled subsystems, and in [109] for optimal tracking.

1.0.1. Major Concerns

The idea of using simple iterative schemes to find the optimal control solution seems very appealing. Also, all schedulers which will be developed in this study are feedback controllers which can be easily used online. However, the major concerns are in terms of stability and convergence. Also, it is desired to discuss different implementation methods and compare their performance. Hence, the major questions are as follows.

- Do the value functions in the iterations converge?
- In case the value functions converge, do they converge to the optimal solution?
- How can we discuss the stability of the system in training?
- How can we find the solution to the policy evaluation in the PI algorithm?
- How does the switching nature of the control problem affects the proposed solutions?
- What are the options to perform the training?

1.0.2. Contributions

The main contributions of this study can be summarized as follows.

- Convergence, stability of evolving control policies, and formulation of a PI algorithm for the switched systems are presented.
- A new approach for proof of convergence of PI algorithm in systems with CT dynamics is presented.

- Derivations for extending the results to optimal tracking controllers in the switched systems are given.
- For online training, training laws based on gradient descent algorithm and recursive least squares algorithm are presented to implement the PI algorithm for the switched systems.
- Concurrent training algorithms with both gradient descent and recursive least squares training laws are presented to implement the PI algorithm for the switched systems.
- A new algorithm is introduced which neglects the policy evaluation and tries to decrease the computational load of PI algorithm.
- A new controller of optimal switching in anti-lock brake system of ground vehicles is introduced which directly addresses the switching nature of the problem.
- The performance of the VI algorithm in optimal switching problems is improved.

1.0.3. Accomplishments

The following publications are resulted from this study.

1- T. Sardarmehni and A. Heydari, **"Sub-optimal Scheduling in Switched Systems with Continuous-time Dynamics: A Gradient Descent Approach"**, Neurocomputing, Volume 285, Pages 10-22, January (2018).

2- T. Sardarmehni and A. Heydari, **"Sub-optimal Scheduling in Switched Systems with Continuous-time Dynamics: A Least Squares Approach"**, IEEE Transactions on Neural Networks and Learning Systems, Volume PP, Number 99, Pages 1-12, September (2017).

3- T. Sardarmehni and A. Heydari, **"Sub-optimal Switching in Anti-lock Brake Systems using Approximate Dynamic Programming"**, Under review at Neurocomputing.

4- T. Sardarmehni and A. Heydari, **"Approximate Solution for Optimal Control of Continuous-time Switched Systems"**, Proceedings of ASME 2016 Dynamic Systems and Control Conference (DSCC2016), Volume 1, Paper No. DSCC2016-9745, Pages V001T02A004-V001T02A014, Minneapolis, MN, October (2016).

5- T. Sardarmehni and A. Heydari, **"Policy Iteration for Optimal Switching with Continuous-time Dynamics"**, Proceedings of 2016 International Joint Conference on Neural Networks (IJCNN), Pages 3536-3543, Vancouver, Canada, July (2016).

6- T. Sardarmehni and A. Heydari, **"Optimal Switching in Anti-lock Brake Systems of Ground Vehicles Based on Approximate Dynamic Programming"**, Proceedings of ASME 2015 Dynamic Systems and Control Conference (DSCC2015), Volume 3, Paper No. DSCC2015-9893, Pages V003T50A010-V003T50A020, Columbus, OH, October (2015).

1.0.4. Structure of the Dissertation

The general structure of the dissertation is as follows. In chapter 2, a classic PI algorithm is formulated for optimal control of switched systems. The stability and convergence of the PI algorithm are investigated. Three learning methods as offline training, online training, and concurrent training are investigated. For online and concurrent training, gradient descent training laws are introduced. In chapter 3, a PI algorithm is introduced and for online and concurrent training, recursive least squares algorithm is used. Also, a new PI algorithm is introduced in this chapter which tries to reduce the computional burden of the PI algorithm during the training. In chapter 4, a VI algorithm is used to derive the optimal control solution for control of anti-lock brake systems in ground vehicles. At last, chapter 5 concludes the dissertation.

Chapter 2

Near-optimal Scheduling in Switched Systems with Continuous-time Dynamics: A Gradient Descent Approach

A feedback solution for approximate optimal scheduling of switched systems with autonomous subsystems and continuous-time dynamics is presented. The proposed solution is based on policy iteration algorithm which provides the optimal switching schedule. Algorithms for offline, online, and concurrent implementation of the proposed solution are presented. For online and concurrent training, gradient descent training laws are used and the performance of the training laws is analyzed. The effectiveness of the presented algorithms is verified through numerical simulations.

Index Terms- optimal control, switched systems, policy iteration, continuous-time dynamics.

2.1. Introduction

In this study, a class of hybrid systems [136] comprised of a finite number of subsystems/modes with continuous-time (CT) dynamics, and a switching rule is considered. In such switched systems, the switching rule assigns the switching instants and active modes [66, 125, 126]. Moreover, the modes are considered to be autonomous which means no continuously varying control exists in the modes [126]. Hence, the only control input is switching among modes. This definition of switched systems embraces many interesting engineering problems [30, 43, 49, 92, 94].

Due to discontinuous nature of the problem, deriving optimal solutions for control of the switched systems is a challenging task [36, 39, 126]. From mathematical point of view, the necessary and sufficient condition for optimality is provided by the underlying Hamilton-Jacobi-Bellman (HJB) equation [58]. However, solving the HJB equation analytically is difficult and generally impossible [116, 117, 122, 123]. The existing solutions for the optimal control problems such as calculus of variations or Dynamic Programming (DP) are intractable in highly nonlinear systems [58, 110] due to *curse of dimensionality* [10, 58, 129]. To remedy the mentioned problem in optimal

control, Approximate Dynamic Programming (ADP) was introduced which approximates the optimal solution [31, 62]. In general, ADP uses function approximators, such as neural networks, to approximate optimal cost-to-go (value function), namely *critic*, and sometimes optimal policy, namely *actor*. The backbone of ADP is application of iterative methods to tune the unknown parameters of the mentioned function approximators in order to approximate the optimal value function which solves the HJB equation [8, 87, 103, 118, 119].

Policy iteration (PI) is an iterative algorithm which is frequently used in the ADP methods to find the optimal solutions [2, 6, 9, 50, 52–54, 77, 110, 114]. In general, the evolving policies generated by PI algorithm are known to be stabilizing [7]. Mathematically, the proof of convergence in online application of PI algorithm in systems with CT dynamics is either based on satisfaction of Persistency of Excitation (PE) condition [9, 20, 110, 120] or application of carefully selected data along with on-trajectory data [50, 52–54, 77]. The latter is called concurrent training and was developed to relax the PE condition since this condition is restrictive and generally cannot be verified online [14].

The ADP-based solutions for optimal control of the switched systems were studied in [36, 39, 40, 88, 91, 134] for discrete-time (DT) dynamics and in [69, 91, 98, 109] for CT dynamics. In [91], the problem of optimal scheduling in the switched systems with CT dynamics and homogenous subsystems was solved with a Value Iteration (VI) algorithm. In [69], an optimal scheduler was developed based on a PI algorithm for switched systems with controlled subsystems where the dynamics of the subsystems include both continuous state and control signals. The presented algorithm trained two different neural networks as actor and critic. In another work, an optimal tracking scheduler was developed in [109]. The proposed scheduler formulated a PI algorithm which also trained two neural networks as actor and critic. The output of the actor was a continuous signal which needed to be discretized for scheduling. Hence, the controller used a hard limiter function which receives the continuous output of the actor and discretizes it, to select the proper mode. In [98], a PI algorithm was developed for optimal scheduling in the switched systems where online training law was derived based on recursive least squares. The comparison between the above-mentioned studies and the current one is given in the sequel.

The challenging nature of the switched system control, lack of sufficient theoretical investigations with ADP methods for this problem in CT dynamics, and the large engineering application of this control problem are the main motivations of this chapter. The main contributions of the present study are as follows[1].

- Convergence, stability of evolving control policies, and formulation of a PI algorithm for the switched systems are presented.
- A new approach for proof of convergence of PI algorithm in systems with CT dynamics is presented.
- Derivations for extending the results to optimal tracking controllers in the switched systems are given.
- A concurrent training algorithm is presented to implement the PI algorithm for the switched systems.

Before presenting the main results, some comparisons between this research and recent relevant studies are given. Compared to [91], the present work deals with general class of switched systems with autonomous subsystems rather than special class of switched systems with homogenous subsystems. Meanwhile, the derivations of the approximate optimal solution in the present study is based on PI algorithm. This is one of the differences between this work and [91] which uses VI as another capable learning algorithm. Compared to [69], the present study deals with switched system with autonomous subsystems where in [69] switched systems with controlled subsystems were studied. Compared to [109], the present study explicitly provides the optimal switching schedule. This scheduling is completely different from the procedure used in [109] for assigning active modes through using a hard limiter function to discretize the output of the actor. At last, the present study uses gradient descent for online training with exponential convergence rate. However, in [98] recursive least squares was used for which the convergence rate is not exponential. Meanwhile, the convergence and stability of the PI algorithm for the switched systems are investigated in this research which were missing in [98].

[1]The preliminary results of this research were presented in 2016 International Joint Conference on Neural Networks (IJCNN 2016) [96].

The rest of the chapter is organized as follows. In section 2.2, notations are introduced. In section 2.3, the optimal switching problem is formulated and the proposed solution is discussed in section 2.4. In section 2.5, the proposed PI algorithm is introduced and its convergence to the optimal solution is analyzed. The implementation of the proposed algorithm with offline, online, and concurrent training methods are discussed in section 2.6 and simulation results are presented in section 2.7. At last, section 2.8, concludes the chapter.

2.2. Notations

Throughout the chapter, $\mathbb{R}$ denotes the real numbers. The set of real n-vectors is denoted with $\mathbb{R}^n$ and set of real $n \times m$ matrices are denoted with $\mathbb{R}^{n\times m}$. The absolute value of a scalar $s \in \mathbb{R}$ is denoted by $|s|$. The Euclidean norm of a vector $v \in \mathbb{R}^n$ is denoted by $\|v\|$ and the 2-norm of a matrix $M \in \mathbb{R}^{n\times m}$ is denoted by $\|M\|$. The transpose operator is denoted by $(.)^T$ and $\lambda_{min}(.)$ represents the minimum eigenvalue of its argument. At last, the gradient of a vector is defined as a column vector and denoted by $\nabla \equiv \frac{\partial}{\partial x}$.

2.3. Problem Formulation

Nonlinear dynamics of a switched system with autonomous subsystems can be presented as

$$\dot{x}(t) = f_\omega(x(t)), \quad \omega \in \mathscr{V} = \{1,2,...,M\}, \quad x(0) = x_0, \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state vector and t denotes the time. The active mode in the system is denoted by subscript ω and $f_\omega : \mathbb{R}^n \to \mathbb{R}^n$ denotes the dynamics of the active mode. Meanwhile, the set of all subsystems/modes that can be selected for the operation of the system is denoted by $\mathscr{V}$ and parameter M is the number of subsystems in the system. Considering $\Omega \subset \mathbb{R}^n$ as the region of interest that includes the origin, it is assumed that each mode $f_\omega(.)$ is *Lipschitz* continuous in Ω and there exists at least one mode for which $f_\omega(0) = 0$. For the operation of the system, the active mode ω may be selected by a feedback control law (scheduler) denoted by $v(.)$, such that at each time t only one mode ω is active.

The selected infinite horizon cost function for the switching problem is given by

$$J(x_0) = \int_0^{\infty} Q(x(\tau))d\tau, \tag{2.2}$$

where $x_0 = x(0)$ and $Q(.) : \mathbb{R}^n \to \mathbb{R}$ is a positive definite function. The objective is finding a stabilizing switching schedule in a feedback form, i.e., $v(x)$, such that the cost function in (2.2) is minimized subject to dynamics presented in (2.1).

2.4. Proposed Solution

Considering (2.2), and a given policy, the cost-to-go, denoted with $V : \mathbb{R}^n \to \mathbb{R}$, can be defined as

$$V(x(t)) = \int_t^{\infty} Q(x(\tau))d\tau. \tag{2.3}$$

Considering a time interval $[t, t+\delta t]$, one has

$$V(x(t)) = \int_t^{t+\delta t} Q(x(\tau))d\tau + V(x(t+\delta t)). \tag{2.4}$$

For notational brevity, hereafter $x = x(t)$ unless otherwise stated. Using the Bellman principle of optimality [58], the optimal cost-to-go, $V^* : \mathbb{R}^n \to \mathbb{R}$, can be defined as

$$V^*(x) = \min_{v(.)} \Big(\int_t^{t+\delta t} Q(x(\tau))d\tau + V^*(x(t+\delta t)) \Big), \tag{2.5}$$

where the policy given by $v(.)$ is exploited to propagate the states from t to $t+\delta t$. As $\delta t \to 0$, the optimal switching policy can be given by

$$v^*(x) = \arg\min_{v \in \mathscr{V}} \Big(\int_t^{t+\delta t} Q(x(\tau))d\tau + V^*(x(t+\delta t)) \Big). \tag{2.6}$$

In order to derive the infinitesimal form of the Bellman equation, the following assumption is required.

Remark 2.4.1 In (2.6), $\arg\min(.)$ operator is used. It is important to note that this operator gives

the mode, i.e. control input, which results in minimum value of its argument and not the minimum value. Also, $v^*(.)$ is the optimal policy, i.e., control law, and it is not a single mode.

Assumption 1 *The value function is continuously differentiable, i.e., $V(.) \in C^1$, [69, 75, 126].*

Remark 2.4.2 Besides serving the purpose in derivation of the desired equations, Assumption 1 makes application of function approximators, such as neural networks, possible in order to *uniformly* approximate the value function [42]. In general, differentiability or even continuity of value functions in optimal control problems is not clear [2]. Differentiability and twice differentiability of the value functions are studied in [4, 12, 26, 28, 102, 112] for some classes of optimal control systems. For Hybrid and switched systems, continuity of the value functions was investigated in [19, 39, 55, 83, 91] for some classes of hybrid and switched systems. Also, differentiability of the value functions in hybrid and switched systems was investigated in [79, 100, 104]. In this chapter we follow the same assumption on the continuous differentiability of the value functions as in [69, 75, 126]. □

The infinitesimal form of (2.4) in CT systems is called Lyapunov equation [2, 110, 113]. Considering Assumption 1, the Lyapunov equation for the switched systems can be defined as

$$Q(x) + (\frac{\partial V(x)}{\partial x})^T f_{v(x)}(x) = 0, \tag{2.7}$$

where $(.)^T$ is the transpose operator and $\frac{\partial}{\partial x}$ is the gradient operator defined as a column vector. Equation (2.7) is an important equation for deriving the approximate solution in the optimal switched systems. Considering the optimal value function $V^*(.)$ and (2.7), one can define the Hamiltonian as

$$H(x, v(.), V_x^*(.)) = Q(x) + (\frac{\partial V^*(x)}{\partial x})^T f_{v(x)}(x). \tag{2.8}$$

The minimizer of the Hamiltonian solves the HJB equation in optimal control problems [58]. Hence, one can define the HJB equation for the switched systems as

$$\min_{v(.)} (H(x, v(.), V_x^*)) = Q(x) + (\frac{\partial V^*(x)}{\partial x})^T f_{v^*(x)}(x) = 0, \tag{2.9}$$

where

$$v^*(x) = \arg\min_{\omega \in \mathscr{V}} \Big((\frac{\partial V^*(x)}{\partial x})^T f_\omega(x) \Big). \tag{2.10}$$

In case $V^*(.)$ is known, one can use the feedback scheduler denoted in (2.10) for real-time scheduling. The inputs of this scheduler are the states of the system and the output is an assigned mode which is the minimizer of the right-hand side of the HJB equation. As one can see from (2.10), the presented scheduler has a simple structure. The challenge here, is how to find the optimal value function, $V^*(.)$, which will be discussed in details in the following sections.

2.5. Proposed PI Algorithm

As mentioned before, ADP methods use iterative schemes for finding the optimal value function. In this study, a PI algorithm is used to find the optimal value function in the switched systems. In order to introduce the main results of the chapter, some preliminary definitions and assumptions are required.

Definition 2.1 A switching policy is admissible if it stabilizes the system presented in (2.1) on Ω and for any $x_0 \in \Omega$, $V(x_0)$ is finite. □

Assumption 2 *There exists at least one admissible policy for the system.*

Assumption 2, it is a controllability-like assumption which guarantees finiteness of the optimal value function.

Iterations in a PI algorithm form two major stages, namely *policy evaluation* and *policy update*. In the policy evaluation stage, performance of a selected policy is evaluated and the value function of that specific policy is calculated. Then, the policy is updated in the policy update stage and the

new policy is used in the next iteration of the PI algorithm. Based on (2.7), the PI algorithm can be formulated as

$$Q(x) + (\frac{\partial V^i(x)}{\partial x})^T f_{v^i(x)}(x) = 0, \tag{2.11}$$

$$v^{i+1}(x) = \arg\min_{\omega \in \mathscr{V}} \Big((\frac{\partial V^i(x)}{\partial x})^T f_\omega(x) \Big). \tag{2.12}$$

The iteration index of the PI algorithm is denoted by superscript i in equations (2.11) and (2.12). Equations (2.11) and (2.12) form the policy evaluation and policy update stages in the PI algorithm, respectively. Considering an arbitrary iteration i of the PI algorithm, the value function of policy $v^i(.)$ is calculated from policy evaluation in (2.11), for all $x \in \Omega$. With the calculated value function of policy $v^i(.)$, policy is updated in policy update stage through (2.12). In summary, one starts with an initial admissible policy $v^0(.)$ and from (2.11), calculates the value function of policy $v^0(.)$, denoted with $V^0(.)$. With $V^0(.)$ and (2.12), one finds the new policy $v^1(.)$. Similarly, with $v^1(.)$ and (2.11), one finds $V^1(.)$. This iterative process continues until no meaningful changes can be detected in the value functions calculated from (2.11).

2.5.1. Convergence and Stability Analysis of the PI Algorithm in the Switched Systems

In this section, stability and convergence of the presented PI algorithm for the switched systems is studied. In order to present the results, some new notations are introduced. We consider value function of a stabilizing policy $h(.)$ as $V_h(.)$. Similarly, we consider the value function at the iteration i of the PI algorithm which is calculated along policy $h(.)$ as $V_h^i(.)$. Also, we consider the state trajectory propagated along policy $h(.)$ as $x^h(.)$. Hereafter, for the sake of notational brevity we refer to $V_{v^i}^i(.)$ as $V^i(.)$ and similarly to $V_{v^{i+1}}^{i+1}(.)$ as $V^{i+1}(.)$.

The stabilizing feature of the evolving policies in the presented PI algorithm is investigated in the following theorem.

Theorem 2.1 *Each immature policy $v^i(\cdot)$ generated by the PI algorithm given by equations (2.11) and (2.12) initiated from an initial admissible policy stabilizes the system in Ω.*

Proof: For proving the stabilizing feature of the evolving policies, one can consider each single value function as a candidate Lyapunov function. Hence, one needs to show that the time derivative

of the chosen candidate Lyapunov function is negative. From the definition of the value function, one has

$$V^i(x) = \int_t^{\infty} Q(x^{v^i}(\tau))d\tau, \tag{2.13}$$

As one can see from (2.13), positive definiteness of $Q(.)$ results in positive definiteness of $V^i(.)$. Also, the value function is continuously differentiable per Assumption 1. Hence, the time derivative of $V^i(.)$ can be defined as

$$\dot{V}^i(x) = (\frac{\partial V^i(x)}{\partial x})^T f_{v^i(x)}(x). \tag{2.14}$$

Replacing the right-hand side of the foregoing equation using (2.11), one has

$$\dot{V}^i(x) = -Q(x) < 0, \tag{2.15}$$

which shows negative definiteness of $\dot{V}^i(.)$. As a result, one can consider $V^i(.)$ as a Lyapunov function, hence, the stability of the system under $v^i(\cdot)$ follows [56]. □

Next, the convergence of the PI algorithm is analyzed.

Lemma 2.2 *Consider $h(.)$ and $g(.)$ as admissible policies. If*

$$\int_t^{t+\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+\delta t)) \leq V_g(x(t)), \tag{2.16}$$

then $V_h(x(t)) \leq V_g(x(t))$.

Proof: The idea for the proof is motivated by analysis of discrete-time systems in [35]. Considering (2.16), for time instant $t+\delta t$ to $t+2\delta t$ one has

$$\int_{t+\delta t}^{t+2\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+2\delta t)) \leq V_g(x^h(t+\delta t)). \tag{2.17}$$

Using (2.17) in (2.16) leads to

$$\int_t^{t+\delta t} Q(x^h(\tau))d\tau + \int_{t+\delta t}^{t+2\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+2\delta t)) \leq V_g(x(t)). \tag{2.18}$$

In other words,

$$\int_t^{t+2\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+2\delta t)) \leq V_g(x(t)). \tag{2.19}$$

Continuing the same procedure as in (2.19), one has

$$\begin{aligned} &\int_t^{t+\delta t} Q(x^h(\tau))d\tau + \int_{t+\delta t}^{t+2\delta t} Q(x^h(\tau))d\tau + \cdots \\ &+ \int_{t+(N-1)\delta t}^{t+N\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+N\delta t)) \leq V_g(x(t)), \end{aligned} \tag{2.20}$$

where N is a positive integer. Combining the integrals leads to

$$\int_t^{t+N\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+N\delta t)) \leq V_g(x(t)). \tag{2.21}$$

As $N \to \infty$, $\lim_{N\to\infty} V_g(x^h(t+N\delta t)) \geq 0$. Therefore,

$$\lim_{N\to\infty} \Big(\int_t^{t+N\delta t} Q(x^h(\tau))d\tau + V_g(x^h(t+N\delta t)) \Big) \geq \int_t^{\infty} Q(x^h(\tau))d\tau \equiv V_h(x(t)). \tag{2.22}$$

Hence, given (2.21) and (2.22), $V_h(x(t)) \leq V_g(x(t))$ which completes the proof. □

Using the results of Lemma 2.2, one can prove the convergence of the PI algorithm.

Theorem 2.3 *The evolving value functions generated from the iterations of the PI algorithm presented by equations (2.11) and (2.12) initiated from an admissible initial policy converge to the optimal value function in* Ω.

Proof: Since $Q(.)$ is not mode dependent, one can consider policy update as

$$v^{i+1}(x) = \arg\min_{\omega\in\mathscr{V}} \Big(Q(x) + (\frac{\partial V^i(x)}{\partial x})^T f_\omega(x) \Big). \tag{2.23}$$

Equations (2.23) and (2.11) lead to

$$Q(x(t)) + (\frac{\partial V^i(x)}{\partial x})^T f_{v^{i+1}(x(t))}(x(t)) \leq 0. \tag{2.24}$$

For any admissible policy $v(.)$, $(\frac{\partial V^i(x(t))}{\partial x})^T f_{v(x(t))}(x(t))$ is the time derivative of $V^i(.)$ calculated along policy $v(.)$. Hence for any admissible policy $v(.)$, one can substitute $\int_t^{t+\delta t}\Big((\frac{\partial V^i(x(\tau))}{\partial x})^T f_{v(x(\tau))} * (x(\tau))\Big)d\tau$ by $V^i(x^v(t+\delta t)) - V^i(x(t))$. Hence, by integrating (2.24) along policy $v^{i+1}(.)$ one has

$$\int_t^{t+\delta t} Q(x^{v^{i+1}}(\tau))d\tau + V^i(x^{v^{i+1}}(t+\delta t)) \leq V^i(x(t)). \tag{2.25}$$

Considering (2.25), using Lemma 2.2 results in $V^{i+1}(x) \leq V^i(x)$ which shows that the sequence of the value functions generated by the PI algorithm are pointwise monotonically non-increasing in Ω.

With the established monotonicity, one can prove the convergence of the PI algorithm as follows. By definition of the optimal value function, $V^*(.) \leq V^i(.)$, $\forall i$, and $\forall x \in \Omega$ because if $V^*(.) > V^i(.)$ then $V^*(.)$ is not the optimal value function. Therefore, the sequence of value functions is lower bounded by $V^*(.)$. The convergence of iterations of the value functions to $V^\infty(.)$ will be resulted directly since any non-increasing sequence of functions which is bounded below converges [93]. Considering $v^\infty(.)$, $V^\infty(.)$, and (2.11) , one can see that $v^\infty(.)$ and $V^\infty(.)$ solve $Q(x) + (\frac{\partial V^\infty(x)}{\partial x})^T f_{v^\infty(x)}(x) = 0$, $\forall t$[1], and $\forall x$. Given (2.12), $\omega = v^\infty(x)$ is the minimizer of $(\frac{\partial V^\infty(x)}{\partial x})^T f_\omega(x)$. Therefore, $\omega = v^\infty(x)$ solves $\min_\omega \Big(Q(x) + (\frac{\partial V^\infty(x)}{\partial x})^T f_\omega(x)\Big) = 0$. In [6], it is proved that the solution for the HJB equation is unique for CT systems. For the switched systems, the HJB equation can be presented as $\min_\omega \Big(Q(x) + (\frac{\partial V^*(x)}{\partial x})^T f_\omega(x)\Big) = 0$. Hence, one can deduct $V^\infty(.) = V^*(.)$ which completes the proof. □

2.6. Implementation

The challenge in implementing the PI algorithm is solving the policy evaluation equation, presented by (2.11), for the evolving value functions. However, solving (2.11) analytically is very difficult. One way to solve (2.11) is training neural networks to approximate the value functions of the selected policies [8, 119].

It is known from Weierstrass approximation theorem that linear-in-parameter neural networks with polynomial basis functions can approximate continuous functions to any degree of precision

[1] Note $x = x(t)$

in a compact set [93]. Also, per Assumption 1, the value function is continuous. Hence, one can define the critic for optimal value function approximation as

$$V^*(x) = W^{*T}\phi(x) + \varepsilon^*(x), \tag{2.26}$$

where $W^* \in \mathbb{R}^m$ is the optimal weight vector, $\phi : \mathbb{R}^n \to \mathbb{R}^m$ is a vector of linearly independent basis functions or polynomials and positive integer m denotes the number of neurons. $\varepsilon^* : \mathbb{R}^n \to \mathbb{R}$ denotes the error of approximating the optimal value function using a finite number of terms. Considering a selected policy $v^i(.)$ and the i^{th} iteration of the PI algorithm, one can define $V^i(.)$ as

$$V^i(x) = W^{iT}\phi(x) + \varepsilon^i(x), \tag{2.27}$$

where $W^i \in \mathbb{R}^m$ is the weight vector and $\varepsilon^i(.)$ is the approximation error of the neural network for approximating value function of policy $v^i(.)$. Substituting (2.27) in (2.11) leads to

$$Q(x) + \left(W^{iT}\nabla\phi(x) + \nabla^T\varepsilon^i(x)\right) f_{v^i(x)}(x) = 0. \tag{2.28}$$

Hence, the residual error of the exact reconstruction, Bellman error, becomes

$$\begin{aligned} \varepsilon_H^i(x) &= Q(x) + W^{iT}\nabla\phi(x) f_{v^i(x)}(x) \\ &= -\nabla^T\varepsilon^i(x) f_{v^i(x)}(x). \end{aligned} \tag{2.29}$$

For approximating $V^i(.)$, one can define $\widehat{V}^i(.)$ as

$$\widehat{V}^i(x) = \widehat{W}_c^{iT}\phi(x). \tag{2.30}$$

In (2.30), $\widehat{W}_c^i \in \mathbb{R}^m$ is the tunable weight vector of the critic. By training, one needs to find $\widehat{W}_c^i$ such that $\widehat{V}^i(x) \approx V^i(x)$ for all $x \in \Omega$. Consequently, the policy evaluation and policy update stages in

the PI algorithm with the proposed value function approximator in (2.30) can be formulated as

$$Q(x) + \widehat{W}_c^{i^T} \nabla\phi(x) f_{v^i(x)}(x) \approx 0, \tag{2.31}$$

$$\hat{v}^{i+1}(x) = \arg\min_{\omega \in \mathscr{V}} \left(\widehat{W}_c^{i^T} \nabla\phi(x) f_\omega(x) \right). \tag{2.32}$$

In general, training can be performed in different fashions, including but not limited to, offline, online and combination of offline and online which is called concurrent training. In what follows, all these training methods are studied. Before presenting the training algorithms, it is worthy of attention that the training methods discussed in the sequel are independent solutions. Based on resource availability and preference, one decides to implement one of them. All the training methods try to find the value function of a selected policy at each iteration of the PI algorithm. In offline training, a random set of training samples is selected offline and this set will not be updated throughout the training process. Hence, this algorithm does not use on-trajectory data. In online training, random states are generated by propagating the states along random switching among modes. For updating the critic, one exploits the existing policy to propagate the states and update the critic weights. Hence, online training uses on-trajectory data for training. At last, in concurrent training both on-trajectory and off-trajectory data are used.

2.6.1. Offline Training (Batch Mode)

With the choice of linear-in-parameter neural networks, as shown in (2.30), one can use least squares method for offline training of the critic in batch mode. When the training process is concluded, the trained critic is used (without re-training) for online scheduling. This process is presented in Algorithm 2.1.

Algorithm 2.1 *Offline Training*

step 1: *Set $i = 0$ and select η random training samples $x^{[l]} \in \Omega$ where $l \in \{1, 2, ..., \eta\}$. Also select a small positive number γ as a convergence tolerance.*

step 2: *Select an initial admissible policy, $\hat{v}^0(.)$.*

step 3: *Substitute the sample states in (2.31).*

step 4: *Find $\widehat{W}_c^i$ from (2.31) using least squares on the entire set of samples.*

step 5: *Update the policy from (2.32) and set $i = i+1$.*

step 6: *If $\|\widehat{W}_c^{i-1} - \widehat{W}_c^i\| \geq \gamma$ go back to step 3.*

step 7: *Set $W^* = \widehat{W}_c^i$ and stop training.*

The flowchart for Algorithm 2.1 is illustrated in Fig 2.1.

Remark 2.6.1 The details of training with least squares method in step 4 of Algorithm 2.1 can be found in Appendix of [40]. □

Remark 2.6.2 The convergence of the PI algorithm in offline training with the presence of approximation errors with least squares is investigated in Theorem 4 of [2]. □

Remark 2.6.3 Offline training for reference tracking in the switched systems is presented in [36] for DT ADP with a VI algorithm. Algorithm 2.1 can be modified to include reference tracking capability. Considering the dynamics of the reference signal as $\dot{r}(t) = f_r(r(t))$ where $r(.) \in \mathbb{R}^n$ denotes the reference signal, and also considering the state as $x(.) \in \mathbb{R}^n$, the cost function, the value function and the policy would be functions of both $x(.)$ and $r(.)$, i.e., $Q = Q(x,r)$, $V = V(x,r)$ and $v = v(x,r)$. Hence, the cost-to-go, $V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, can be represented as[1]

$$V(x,r) = \int_t^\infty Q(x(\tau), r(\tau)) d\tau, \tag{2.33}$$

where $Q : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a positive definite function. Similarly the optimal value function, $V^* : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, becomes

$$V^*(x,r) = \min_{v(.,.)} \Big(\int_t^\infty Q(x(\tau), r(\tau)) d\tau \Big). \tag{2.34}$$

[1] $r = r(t)$ unless otherwise stated.

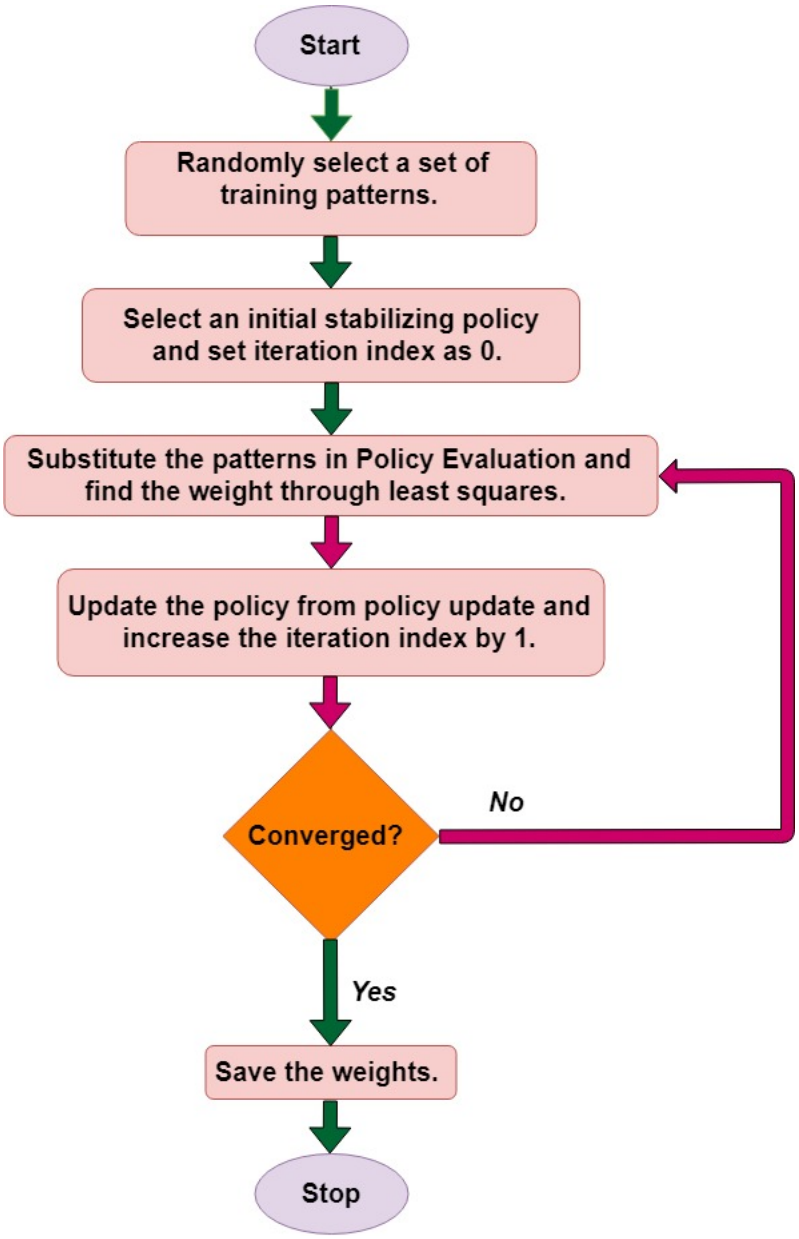

Figure 2.1: Flowchart for offline training with PI algorithm.

Considering the value function of a selected policy as $V(.,.)$, the Bellman equation can be presented as

$$V(x,r) = \int_t^{t+\delta t} Q(x(\tau), r(\tau))d\tau + V(x(t+\delta t), r(t+\delta t)). \tag{2.35}$$

Hence, the infinitesimal form of (2.35) becomes[1]

$$Q(x,r) + (\frac{\partial V(x,r)}{\partial x})^T f_{v(x,r)}(x) + (\frac{\partial V(x,r)}{\partial r})^T f_r(r) = 0. \tag{2.36}$$

Value function approximation can be preformed through critic network which should be modified to have two inputs as $x(.)$ and $r(.)$. Hence, $\widehat{V}(x,r) = \widehat{W}_c^T \phi(x,r)$ where $\phi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ is the vector of linearly independent basis functions and positive integer m is the number of neurons. With this critic, the policy evaluation stage in the PI algorithm for reference tracking can be formulated as

$$Q(x,r) + \widehat{W}_c^{i^T} \left(\frac{\partial \phi(x,r)}{\partial x} f_{\hat{v}^i(x,r)}(x) + \frac{\partial \phi(x,r)}{\partial r} f_r(r) \right) = 0. \tag{2.37}$$

Similarly, the policy update can be represented as

$$v^{i+1}(x,r) = \arg\min_{\omega \in \mathscr{V}} \left(\widehat{W}_c^{i^T} \frac{\partial \phi(x,r)}{\partial x} f_\omega(x) \right). \tag{2.38}$$

The modified algorithm for reference tracking can be implemented as Algorithm 2.2.

Algorithm 2.2 *Offline Training for Reference Tracking*

step 1: *Set $i = 0$. Select η random state training patterns $x^{[l]} \in \Omega$ and η random samples for reference signal $r^{[l]} \in \Omega$ where $l \in \{1,2,...,\eta\}$. Also select a small positive number γ as a convergence tolerance.*

step 2: *Select an initial admissible policy, $\hat{v}^0(.,.)$.*

step 3: *Substitute all sample states and reference signals in (2.37).*

[1]Note that: $V(x(t+\delta t), r(t+\delta t)) = V(x(t), r(t)) + \frac{\partial V}{\partial x}^T (x(t+\delta t) - x(t)) + \frac{\partial V}{\partial r}^T (r(t+\delta t) - r(t))$. By letting $\delta t \to 0$, one can derive (2.36) from (2.35).

step 4: *Find $\widehat{W}_c^i$ from (2.37) using least squares on the entire set of samples.*

step 5: *Update the policy from (2.38) and set $i = i+1$.*

step 6: *If $\|\widehat{W}_c^{i-1} - \widehat{W}_c^i\| \geq \gamma$ go back to step 3.*

step 7: *Set $W^* = \widehat{W}_c^i$ and stop training.*

□

2.6.2. Online Training

In online training, the weight vector of the critic is updated online with sequential data and the state is propagated simultaneously with the iterations of the PI algorithm using the evolving policies. The basic idea applied in this section for online training is using an inner loop in which the value function of a selected policy is calculated by using a learning law derived from a gradient descent algorithm.

Let the current estimate of $\widehat{W}_c^i$ at time instant t be denoted by $\widehat{W}^i(t)$. One can define the instantaneous error as

$$e(t) = Q(x) + \widehat{W}^{i^T}(t)\nabla\phi(x)f_{\hat{v}^i(x)}(x). \tag{2.39}$$

The goal of the online training is to find $\widehat{W}^i(.)$ such that the squared residual error defined as

$$E(t) = \frac{1}{2}e^T(t)e(t) \tag{2.40}$$

is minimized. For minimization of (2.40), normalized gradient descent algorithm [110] is used as

$$\begin{aligned}\dot{\widehat{W}}^i(t) &= -\alpha\frac{\partial E(t)}{\partial \widehat{W}^i(t)} \\ &= -\alpha\frac{\sigma^i(t)}{(\sigma^{i^T}(t)\sigma^i(t)+1)^2}\Big(\widehat{W}^{i^T}(t)\sigma^i(t) + Q(x)\Big),\end{aligned} \tag{2.41}$$

where $\alpha \in \mathbb{R}$ is a positive number called learning rate and $\sigma^i(t) = \nabla\phi(x(t))f_{\hat{v}^i\left(x(t)\right)}(x(t))$. The weight error is defined as $\widetilde{W}^i(t) = W^i - \widehat{W}^i(t)$. Noting that W^i is time invariant, the dynamics of the

error can be found by taking the time derivative of $\widetilde{W}^i(t)$ as

$$\begin{aligned}\dot{\widetilde{W}}^i(t) &= -\dot{\widehat{W}}^i(t)\\ &= \alpha\frac{\sigma^i(t)}{(\sigma^{i^T}(t)\sigma^i(t)+1)^2}\Big(\widehat{W}^{i^T}(t)\sigma^i(t)+Q(x)\Big).\end{aligned} \tag{2.42}$$

Substituting $\widehat{W}^i(t)$ with $W^i-\widetilde{W}^i(t)$ and $W^{i^T}\sigma^i(t)+Q(x)$ by $\varepsilon_H^i(x)$, one has

$$\begin{aligned}\dot{\widetilde{W}}^i(t) &= \alpha\frac{\sigma^i(t)}{(\sigma^{i^T}(t)\sigma^i(t)+1)^2}\Big(-\widetilde{W}^{i^T}(t)\sigma^i(t)+\varepsilon_H^i(x)\Big)\\ &= -\alpha\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t)\widetilde{W}^i(t)+\alpha\bar{\sigma}^i(t)\frac{\varepsilon_H^i(x)}{m_s(t)}.\end{aligned} \tag{2.43}$$

In (2.43), $\bar{\sigma}^i(t)=\frac{\sigma^i(t)}{(1+\sigma^{i^T}(t)\sigma^i(t))}$ and $m_s(t)=(1+\sigma^{i^T}(t)\sigma^i(t))$, motivated by [110].

The online training algorithm can be formulated now. In online training, there are two loops. The inner loop in which the value function of a selected policy is calculated for all $x\in\Omega$. At each iteration of the outer loop, the policy is updated based on the value function of the existing policy. To ensure the convergence of the weights in the inner loop, PE condition is used in the inner loop which can be defined as follows.

Definition 2.2 Signal $\bar{\sigma}^i(t)$ is persistently excited if for all t there exist constants $\beta_1>0$, $\beta_2>0$ and $\delta t>0$, such that

$$\beta_1\mathbf{I}\leq\int_t^{t+\delta t}\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)d\tau\leq\beta_2\mathbf{I}, \tag{2.44}$$

where $\mathbf{I}$ denotes the identity matrix of the proper dimensions [110]. □

In systems with control affine dynamics, PE condition can be satisfied by adding a probing noise signal to the inputs of the system [9, 50, 53, 54, 77, 110]. In case of the switched systems, such inputs are not available and PE condition can be satisfied through random switching. Hence, one can select a random mode and propagate the states along that mode for a certain duration of time to enforce the PE condition. At this point, the online training algorithm can be formulated as Algorithm 2.3.

Algorithm 2.3 *Online Training*

step 1: *Set $i = 0$ and select an initial admissible policy $\hat{v}^0(.)$. Select a region of training Ω, and a random initial weight vector $\widehat{W}^0(0) \in \mathbb{R}^m$, where m is the number of neurons. Measure the system's initial state $x_0 \in \Omega$. Also, select a positive real number α as the learning rate. Moreover, select two time intervals as δt_1 and δt_2, and two small positive values γ_1 and γ_2 as convergence tolerances. At last, set $\breve{W}^1 = \widehat{W}^0(0)$ and $\zeta = 1$.*

step 2: *Select a random mode and let the system run with that mode for duration of δt_1, i.e., operate the system.*

step 3: *Conduct the following inner loop:*

step 3.1: Use $\hat{v}^i(.)$ and run the system with it.

step 3.2: Update $\widehat{W}^i(.)$ using (2.41). Then store $\widehat{W}^i(.)$ as $\breve{W}^{\zeta+1}$ and set $\zeta = \zeta + 1$.

step 3.3: If $\|\breve{W}^{\zeta} - \breve{W}^{\zeta-1}\| \geq \gamma_1$, check the elapsed time from beginning of step 3. If the elapsed time is less than δt_2, go back to step 3.1. Otherwise, go back to step 2.

step 4: *Set $\widehat{W}_c^i = \widehat{W}^i(.)$. If $\|\widehat{W}_c^i - \widehat{W}_c^{i-1}\| \geq \gamma_2$ (for $i \geq 1$), update the policy, $\hat{v}^{i+1}(.)$, from Eq. (2.32), set $\breve{W}^1 = \widehat{W}_c^i$, $\zeta = 1$, $P(.) = \alpha_0 \mathbf{I}$, $i = i+1$ and go back to step 2. Otherwise, set $W^* = \widehat{W}_c^i$ and stop training.*

The flowchart of Algorithm 2.3 is shown in Fig. 2.2.

Before analyzing the online training algorithm, the following assumptions are required.

Assumption 3 *The set of basis functions and their gradients are bounded in the compact region of training Ω, i.e., $\forall x \in \Omega$, $\|\phi(x)\| \leq \phi_{max}$ and $\|\nabla\phi(x)\| \leq \phi_{\nabla,max}$.*

Assumption 4 *The neural network approximation error, $\varepsilon^i(.)$, and the Bellman residual error, $\varepsilon_H^i(.)$, are bounded in the compact region of training Ω, i.e., $\forall x \in \Omega$, $|\varepsilon^i(x)| \leq \varepsilon_{max}$, $|\varepsilon_H^i(x)| \leq \varepsilon_{H,max}$.*

Now, one is ready to establish an upper bound for the error as in the following lemma.

Lemma 2.4 *Consider the error dynamics as (2.43). Let signal $\bar{\sigma}^i(.)$ be PE and Assumptions 1-4 hold. As $t \to \infty$, an upper bound of the error signal $\|\widetilde{W}^i(t)\|$ will converge exponentially to $\sqrt{\frac{2\alpha C}{1-\Gamma}}$ where C and Γ are constants and α is the learning rate.*

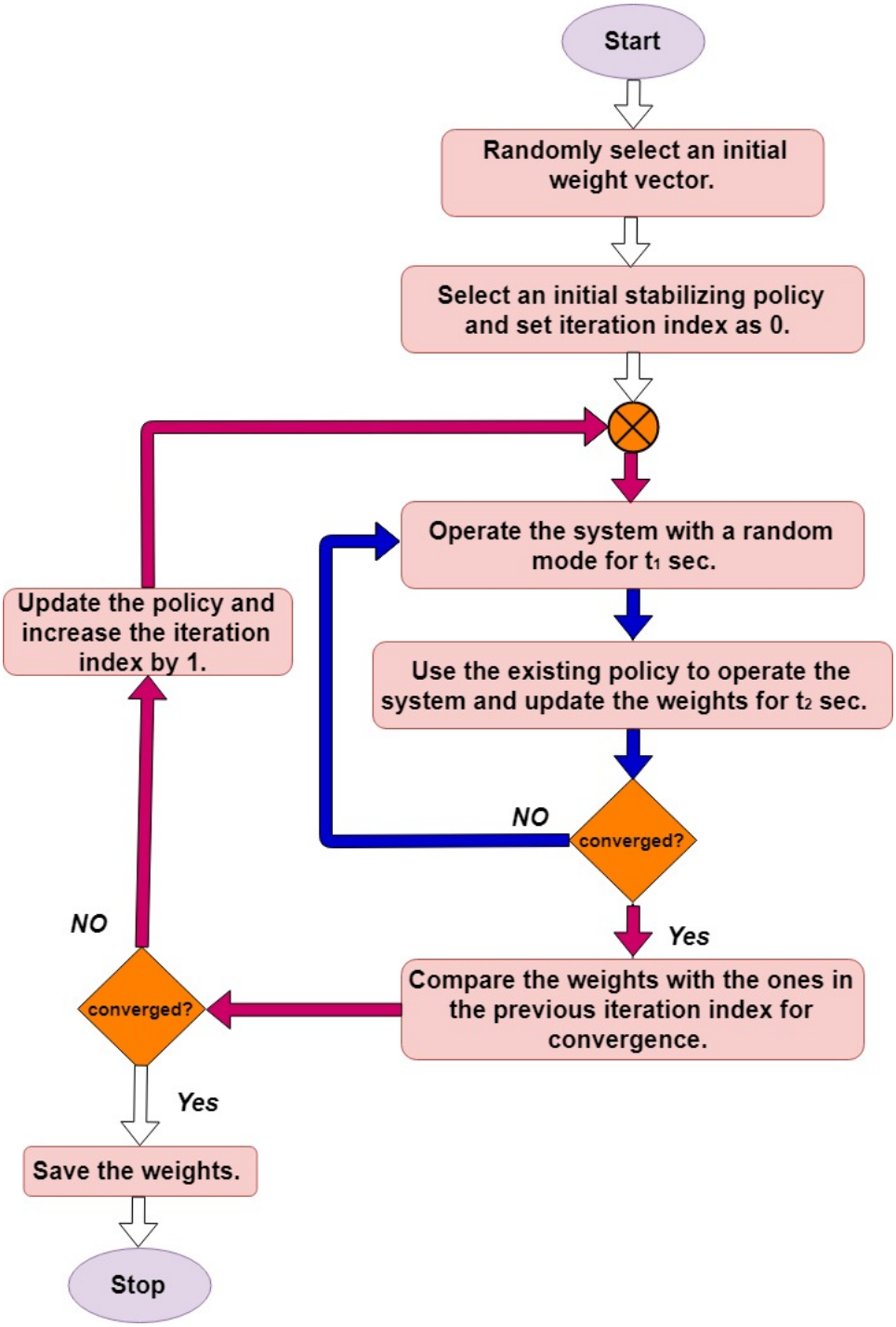

Figure 2.2: Flowchart for online training with PI algorithm.

Proof: The proof is an extended version of the one presented in [45]. Consider the following Lyapunov function candidate

$$L(t) = \frac{1}{2}\alpha^{-1}\widetilde{W}^{i^T}(t)\widetilde{W}^{i}(t), \tag{2.45}$$

where $\widetilde{W}^{i}(t) = \widetilde{W}^{i}(t)$. Taking the time derivative of the (2.45), one has

$$\dot{L}(t) = \alpha^{-1}\widetilde{W}^{i^T}(t)\dot{\widetilde{W}}^{i}(t). \tag{2.46}$$

Substituting $\dot{\widetilde{W}}^{i}(.)$ from (2.43) leads to

$$\dot{L}(t) = -\widetilde{W}^{i^T}(t)\bar{\sigma}^{i}(t)\bar{\sigma}^{i^T}(t)\widetilde{W}^{i}(t) + \widetilde{W}^{i^T}(t)\bar{\sigma}^{i}(t)\frac{\varepsilon_H^i(x)}{m_s(t)}. \tag{2.47}$$

In (2.47), $\bar{\sigma}^{i}(.) = \frac{\sigma^{i}(.)}{m_s(.)}$. Integrating (2.47) from t to $t+\delta t$, one has

$$\begin{aligned} L(t+\delta t) - L(t) = &-\int_t^{t+\delta t} \widetilde{W}^{i^T}(\tau)\bar{\sigma}^{i}(\tau)\bar{\sigma}^{i^T}(\tau)\widetilde{W}^{i}(\tau)d\tau \\ &+\int_t^{t+\delta t} \widetilde{W}^{i^T}(\tau)\bar{\sigma}^{i}(\tau)\frac{\varepsilon_H^i(x)}{m_s(\tau)}d\tau. \end{aligned} \tag{2.48}$$

In order to continue the proof, one needs to first verify the integrability of integrands on the right-hand side of (2.48). Due to existence of $\bar{\sigma}^{i}(.)$ in the integrands, these functions might be discontinuous because of probable switching and integrability is not clear. Noting that the number of switching is countable[1] [132], the integrands, for any admissible policy, are piecewise continuous functions with countable number of first kind [93] discontinuities. The Lebesgue theorem states that discontinuous functions are integrable in Riemann sense if and only if they are bounded and the set of discontinuities forms a set of measure zero [1]. Boundedness of the terms in the integrand of (2.48) can be established as follows. Considering $L(t)$ as a candidate Lyapunov function in (2.45), $\dot{L}(t)$ can be formulated as (2.47). By applying norms on the right-hand side of (2.47), it is easy to see $\dot{L}(t) \leq \|\widetilde{W}^{i^T}(t)\bar{\sigma}^{i}(t)\|(\varepsilon_{H,max} - \|\widetilde{W}^{i^T}(t)\bar{\sigma}^{i}(t)\|)$. This results in boundedness of $\widetilde{W}^{i^T}(t)\bar{\sigma}^{i}(t)$.

[1] It is worthy of attention that even if 'Zeno' happens, i.e., infinite number of switching in a finite time [48, 132], the number of switching will still be countable [48, 132]. This countability also holds in Fuller's phenomenon in optimal control problems [11, 27] which is considered as an equivalent phenomenon to Zeno executions in hybrid systems [59, 64].

Also per Assumption 4, $\varepsilon_H^i(.)$ is bounded which leads to the boundedness of the second integrand in (2.48). Meanwhile, considering countable number of switching, one can see that the sets of discontinuities in the integrands are sets of measure zero. Hence, the integrands are integrable in Riemann sense. Considering the right-hand side of (2.48), it is clear that

$$\begin{aligned}&\int_t^{t+\delta t} \widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)\widetilde{W}^i(\tau)d\tau\\&=\int_t^{t+\delta t}(\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau))^2 d\tau.\end{aligned} \tag{2.49}$$

Equation (2.49) has the same structure of (4.8.35) in [45]. As mentioned in [45], by adding and subtracting $\widetilde{W}^i(t)$ one has $\widetilde{W}^i(\tau)=\widetilde{W}^i(t)+(\widetilde{W}^i(\tau)-\widetilde{W}^i(t))$. Hence, one has

$$\begin{aligned}&\int_t^{t+\delta t}(\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau))^2 d\tau\\&=\int_t^{t+\delta t}\left(\widetilde{W}^{i^T}(t)\bar{\sigma}^i(\tau)+(\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)\right)^2 d\tau.\end{aligned} \tag{2.50}$$

Using the property that $(x+y)^2\geq\frac{1}{2}x^2-y^2$ [45], it follows that

$$\begin{aligned}&\int_t^{t+\delta t}\left(\widetilde{W}^{i^T}(t)\bar{\sigma}^i(\tau)+(\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)\right)^2 d\tau\\&\geq\int_t^{t+\delta t}\left(\frac{1}{2}(\widetilde{W}^{i^T}(t)\bar{\sigma}^i(\tau))^2-\left((\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)\right)^2\right)d\tau\\&\geq\frac{1}{2}\int_t^{t+\delta t}\widetilde{W}^{i^T}(t)\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)\widetilde{W}^i(t)d\tau\\&-\int_t^{t+\delta t}\left((\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)\right)^2 d\tau.\end{aligned} \tag{2.51}$$

Considering $\int_t^{t+\delta t}\widetilde{W}^{i^T}(t)\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)\widetilde{W}^i(t)d\tau$, since $\widetilde{W}^i(t)$ is not function of τ it can commute out of integral. Also, from Definition 2.2, $\beta_1\mathbf{I}\leq\int_t^{t+\delta t}\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)d\tau$. Therefore,

$$
\begin{aligned}
&\widetilde{W}^{i^T}(t)\Big(\int_t^{t+\delta t}\bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)d\tau\Big)\widetilde{W}^i(t)\\
&\geq \widetilde{W}^{i^T}(t)\beta_1\mathbf{I}\widetilde{W}^i(t)\\
&\geq 2\alpha\beta_1 L(t).
\end{aligned}
\tag{2.52}
$$

With the same procedure explained in [45], one has $\widetilde{W}^i(\tau)-\widetilde{W}^i(t)=\int_t^\tau \dot{\widetilde{W}}^i(z)dz$. By substituting for $\dot{\widetilde{W}}(z)$ from (2.43) and expanding the integrals one can pursue as

$$
\begin{aligned}
\widetilde{W}^i(\tau)-\widetilde{W}^i(t)=&\int_t^\tau \alpha\bar{\sigma}^i(z)\frac{\varepsilon_H^i(x)}{m_s(z)}dz\\
&-\int_t^\tau \alpha\bar{\sigma}^i(z)\widetilde{W}^T(z)\bar{\sigma}^i(z)dz.
\end{aligned}
\tag{2.53}
$$

Applying the transpose operator and multiplying both sides of (2.53) by $\bar{\sigma}^i(\tau)$, one has

$$
\begin{aligned}
(\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)=&\int_t^\tau \alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\\
&-\int_t^\tau \alpha\bar{\sigma}^{i^T}(z)\widetilde{W}(z)\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau)dz.
\end{aligned}
\tag{2.54}
$$

Through some algebraic manipulations and using the square function it follows that

$$
\begin{aligned}
&\Big((\widetilde{W}^i(\tau)-\widetilde{W}^i(t))^T\bar{\sigma}^i(\tau)\Big)^2=\Big(\int_t^\tau \alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\\
&-\int_t^\tau \alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2.
\end{aligned}
\tag{2.55}
$$

Noting that $(x-y)^2 \leq 2x^2+2y^2$, one has

$$\begin{aligned}
&\int_t^{t+\delta t}\Big(\int_t^{\tau}\alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz \\
&-\int_t^{\tau}\alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2 d\tau \\
&\leq \int_t^{t+\delta t}\Bigg(2\Big(\int_t^{\tau}\alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\Big)^2 \\
&+2\Big(\int_t^{\tau}\alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2\Bigg)d\tau.
\end{aligned} \tag{2.56}$$

As one can see the right-hand side of inequality (2.56) has two integrals as outer integral with integration variable τ and the inner integral with integration variable z. Considering the outer integral, one can use the linear property of integrals to show that $\int((\int f(x)dz)^2+(\int g(x)dz)^2)d\tau = \int(\int f(x)dz)^2d\tau+\int(\int g(x)dz)^2d\tau$. Considering $f(x)=h(x)k(x)$, from Cauchy-Schwartz inequality one can show that $(\int f(x)dz)^2 \leq \int(h(x))^2dz\int(k(x))^2dz$. Using these properties, one can consider each squared (inner) integral on the right-hand side of inequality (2.56) as follows.

$$\begin{aligned}
&\Big(\int_t^{\tau}\alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\Big)^2 \\
&\leq \int_t^{\tau}(\alpha\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))^2dz\int_t^{\tau}(\frac{\varepsilon_H^i(x)}{m_s(z)})^2dz.
\end{aligned} \tag{2.57}$$

Using norms on the right-hand side of inequality (2.57) leads to

$$\begin{aligned}
&\int_t^{\tau}(\alpha\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))^2dz\int_t^{\tau}(\frac{\varepsilon_H^i(x)}{m_s(z)})^2dz \\
&\leq \int_t^{\tau}\alpha^2\|\bar{\sigma}^i(z)\|^2\|\bar{\sigma}^i(\tau)\|^2dz\int_t^{\tau}|\frac{\varepsilon_H^i(x)}{m_s(z)}|^2dz \\
&\leq \int_t^{\tau}\alpha^2dz\int_t^{\tau}\varepsilon_{H,max}^2dz \\
&\leq \alpha^2\varepsilon_{H,max}^2(\tau-t)^2.
\end{aligned} \tag{2.58}$$

This is an upper bound for the first inner integral on the right-hand side of inequality (2.56). In deriving the upper bounds as in inequality (2.58), it may be noted that $\bar{\sigma}^i(.)$ is normalized hence $\|\bar{\sigma}^i(.)\|\leq 1$. Meanwhile, since $m_s(.)\geq 1$, $|\frac{\varepsilon_H^i(.)}{m_s(.)}|\leq \varepsilon_{H,max}$.

Considering the second inner integral on the right-hand side of inequality (2.56), form Cauchy-Schwartz inequality it follows that [45]

$$\begin{aligned}
&\Big(\int_t^\tau \alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2 \\
&\leq (\int_t^\tau (\alpha\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)(\int_t^\tau (\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))^2 dz) \\
&\leq (\int_t^\tau (\alpha\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)(\int_t^\tau (\|\bar{\sigma}^i(z)\|\|\bar{\sigma}^i(\tau)\|)^2 dz) \\
&\leq (\int_t^\tau (\alpha\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)(\int_t^\tau dz) \\
&\leq (\tau-t)\int_t^\tau (\alpha\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz.
\end{aligned} \tag{2.59}$$

Considering the upper bounds developed in inequality (2.58) and (2.59) and inequality (2.56), one has

$$\begin{aligned}
&\int_t^{t+\delta t}\Big(2\Big(\int_t^\tau \alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\Big)^2 \\
&+2\Big(\int_t^\tau \alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2\Big)d\tau \\
&\leq 2\alpha^2\varepsilon_{H,max}^2\int_t^{t+\delta t}(\tau-t)^2 d\tau \\
&+2\alpha^2\int_t^{t+\delta t}(\tau-t)(\int_t^\tau (\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)d\tau \\
&=2\alpha^2\varepsilon_{H,max}^2\frac{(\delta t)^3}{3}+2\alpha^2\int_t^{t+\delta t}(\tau-t)(\int_t^\tau (\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)d\tau.
\end{aligned} \tag{2.60}$$

Changing the sequence of integration [45] in the last term of inequality (2.60) leads to

$$\begin{aligned}
&\int_t^{t+\delta t}(\tau-t)(\int_t^\tau (\widetilde{W}^T(z)\bar{\sigma}^i(z))^2 dz)d\tau \\
&=\int_t^{t+\delta t}(\widetilde{W}^T(z)\bar{\sigma}^i(z))^2(\int_z^{t+\delta t}(\tau-t)d\tau)dz \\
&=\int_t^{t+\delta t}(\widetilde{W}^T(z)\bar{\sigma}^i(z))^2(\frac{\delta t^2}{2}-\frac{(t-z)^2}{2})dz \\
&\leq\int_t^{t+\delta t}(\widetilde{W}^T(z)\bar{\sigma}^i(z))^2(\frac{\delta t^2}{2})dz.
\end{aligned} \tag{2.61}$$

Substituting inequality (2.61) in inequality (2.60), through some algebraic manipulations one has

$$\begin{aligned}
&\int_t^{t+\delta t}\Bigg(2\Big(\int_t^{\tau}\alpha\bar{\sigma}^{i^T}(z)\frac{\varepsilon_H^i(x)}{m_s(z)}\bar{\sigma}^i(\tau)dz\Big)^2\\
&+2\Big(\int_t^{\tau}\alpha(\widetilde{W}^T(z)\bar{\sigma}^i(z))(\bar{\sigma}^{i^T}(z)\bar{\sigma}^i(\tau))dz\Big)^2\Bigg)d\tau\\
&\leq \alpha^2(\delta t)^2(\varepsilon_{H,max}^2\frac{2\delta t}{3}+\int_t^{t+\delta t}(\widetilde{W}^T(z)\bar{\sigma}^i(z))^2dz).
\end{aligned} \tag{2.62}$$

Using the results developed in inequality (2.52) and (2.62) in inequality (2.50) and (2.51), it follows

$$\begin{aligned}
&\int_t^{t+\delta t}(\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau))^2d\tau \geq \alpha\beta_1 L(t)-\frac{2}{3}\alpha^2\varepsilon_{H,max}^2(\delta t)^3\\
&-\alpha^2(\delta t)^2\int_t^{t+\delta t}(\widetilde{W}^T(z)\bar{\sigma}^i(z))^2dz.
\end{aligned} \tag{2.63}$$

Some algebraic manipulations leads to

$$\begin{aligned}
&\int_t^{t+\delta t}(\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau))^2d\tau\\
&\geq \frac{1}{1+\alpha^2(\delta t)^2}\Big(\alpha\beta_1 L(t)-\frac{2}{3}\alpha^2\varepsilon_{H,max}^2(\delta t)^3\Big).
\end{aligned} \tag{2.64}$$

Using the results of inequality (2.64) in inequality (2.48) and (2.49), one has

$$\begin{aligned}
&L(t+\delta t)-L(t)\leq \frac{-1}{1+\alpha^2(\delta t)^2}\Big(\alpha\beta_1 L(t)-\frac{2}{3}\alpha^2\varepsilon_{H,max}^2(\delta t)^3\Big)\\
&+\int_t^{t+\delta t}\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau)\frac{\varepsilon_H^i}{m_s}d\tau.
\end{aligned} \tag{2.65}$$

Defining $\Gamma_0 = \frac{1}{1+\alpha^2(\delta t)^2}$, by using norm operator on the right-hand side of inequality (2.65), it follows that

$$\begin{aligned} L(t+\delta t) - L(t) &\leq -\Gamma_0 \alpha \beta_1 L(t) + \Gamma_0 \frac{2}{3} \alpha^2 \varepsilon_{H,max}^2 (\delta t)^3 \\ &+ \int_t^{t+\delta t} \|\widetilde{W}^{i^T}(\tau) \bar{\sigma}^i(\tau)\| |\frac{\varepsilon_H^i}{m_s}| d\tau \\ &\leq -\Gamma_0 \alpha \beta_1 L(t) + \Gamma_0 \frac{2}{3} \alpha^2 \varepsilon_{H,max}^2 (\delta t)^3 \\ &+ \varepsilon_{H,max} \int_t^{t+\delta t} \|\widetilde{W}^{i^T}(\tau) \bar{\sigma}^i(\tau)\| d\tau. \end{aligned} \tag{2.66}$$

Since $\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau)$ is bounded, one has $\int_t^{t+\delta t} \|\widetilde{W}^{i^T}(\tau)\bar{\sigma}^i(\tau)\| d\tau \leq C_1$ where C_1 is a positive constant. Through some algebraic manipulations, one can show

$$\begin{aligned} L(t+\delta t) \leq & (1 - \Gamma_0 \alpha \beta_1) L(t) + \Gamma_0 \frac{2}{3} \alpha^2 \varepsilon_{H,max}^2 (\delta t)^3 \\ & + \varepsilon_{H,max} C_1. \end{aligned} \tag{2.67}$$

Noting $\Gamma_0 \frac{2}{3} \alpha^2 \varepsilon_{H,max}^2 (\delta t)^3 = \frac{\frac{2}{3}\alpha^2 \varepsilon_{H,max}^2 (\delta t)^3}{1+\alpha^2(\delta t)^2}$, it is easy to see $\Gamma_0 \frac{2}{3} \alpha^2 \varepsilon_{H,max}^2 (\delta t)^3 \leq \delta t \varepsilon_{H,max}^2$. Letting $C = \delta t \varepsilon_{H,max}^2 + \varepsilon_{H,max} C_1$, and $\Gamma = 1 - \alpha \Gamma_0 \beta_1$, it follows that

$$L(t+\delta t) \leq \Gamma L(t) + C. \tag{2.68}$$

Considering $t = (n-1)\delta t$, one can rewrite inequality (2.68) as

$$L(n\delta t) \leq \Gamma L((n-1)\delta t) + C. \tag{2.69}$$

Now, consider the following sequence

$n = 1 \Rightarrow L(\delta t) \leq \Gamma L(0) + C,$

$n = 2 \Rightarrow L(2\delta t) \leq \Gamma L(\delta t) + C = \Gamma^2 L(0) + \Gamma C + C.$

Continuing the same procedure, one can see that

$$L(n\delta t) \leq \Gamma^n L(0) + \sum_{k=1}^{n} \Gamma^{k-1} C. \tag{2.70}$$

From PE condition, one has $\beta_1 \mathbf{I} \leq \int_t^{t+\delta t} \bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)d\tau$. Since $\int_t^{t+\delta t} \bar{\sigma}^i(\tau)\bar{\sigma}^{i^T}(\tau)d\tau$ is a symmetric positive definite matrix, one can see that the norm of it is equal to its largest singular value (which is the largest eigenvalue). Hence, by applying norms, one has $\beta_1 \leq \int_t^{t+\delta t} \|\bar{\sigma}^i(\tau)\|^2 d\tau \leq \int_t^{t+\delta t} d\tau$ which results in $\beta_1 \leq \delta t$. This shows that $\alpha\beta_1\Gamma_0 \leq \frac{\alpha\delta t}{(1+\alpha^2(\delta t)^2)} < 1$. Considering $\Gamma = 1 - \alpha\beta_1\Gamma_0$, it can be seen that $0 < \Gamma < 1$. Hence, $\sum_{k=1}^{n} \Gamma^{k-1}$ forms a geometric series [93] and one can rewrite inequality (2.70) as

$$L(n\delta t) \leq \Gamma^n L(0) + C\frac{1-\Gamma^n}{1-\Gamma}. \tag{2.71}$$

From inequality (2.71), One can see that as $n \to \infty$, $\frac{1-\Gamma^n}{1-\Gamma} \to \frac{1}{1-\Gamma}$ and $\Gamma^n L(0) \to 0$ exponentially with decay factor of $\frac{-1}{\delta t}\ln\Gamma$. Hence, as $n \to \infty$, $L(n\delta t) \leq \frac{C}{1-\Gamma}$. At sufficiently large n, $L(n\delta t) \approx L(t)$ where from (2.45), $L(t) = \frac{1}{2}\alpha^{-1}\widetilde{W}^{i^T}(t)\widetilde{W}^i(t) = \frac{1}{2}\alpha^{-1}\|\widetilde{W}^i(t)\|^2$. Taking the square root leads to

$$\|\widetilde{W}^i(t)\| \leq \sqrt{\frac{2\alpha C}{1-\Gamma}}. \tag{2.72}$$

It is important to note that variations of Γ can be regulated by choice of α which means that one can prevent the denominator on the right-hand side of inequality (2.72) from becoming too small. Meanwhile, C in the numerator of inequality (2.72) is linearly linked to $\varepsilon_{H,max}$ which shows that the upper bound in (2.72) becomes smaller as the approximation precision of the neural networks improves. □

2.6.3. Sequential Offline Training (Gradient Descent)

Sequential offline training can be formulated based on online training law presented in (2.41). The proof for convergence of sequential offline training is the same as that of the online training which was discussed in details in online training section. In sequential offline training, random training patterns, instead of random modes, are used individually in the inner loop. It is worthy

of attention that the sequential offline training is an offline process. Hence, unlike online training, system is not operated during the training process and therefore stability of the system during the training process is not an issue. Meanwhile, the choice of random states in the inner loop can fulfill the PE condition. Since the choice of random states bypasses the dynamics of the system, a slight change is recommended in the training law as follows

$$\dot{\widehat{W}}^{i}(t) = -\alpha \frac{\sigma^{i}(t)}{\left(\sigma_{max}^{T}\sigma_{max}+1\right)^{2}}\left(\widehat{W}^{i^{T}}(t)\sigma^{i}(t)+Q(x)\right), \tag{2.73}$$

where $\sigma_{max} = \max_{x\in\Omega}\left(\nabla\phi(x) f_{\hat{v}^{i}(x)}(x)\right)$. The sequential offline training algorithm can be implemented as Algorithm 2.4.

Algorithm 2.4 *Sequential Offline Training*

step 1: *Pick an initial admissible policy, $\hat{v}^{0}(.)$, an initial state x_0, and two small positive numbers as the convergence tolerance γ_1 and γ_2. Also, set $i = 0$, and $\zeta = 1$. At last, select a random initial weight vector $\widehat{W}^{0}(0) \in \mathbb{R}^{m}$, where m is the number of neurons and set $\breve{W}^{1} = \widehat{W}^{i}(0)$.*

step 2: *Conduct the following inner loop:*

step 2-1: Select a random state x.

step 2-2: Use $\hat{v}^{i}(.)$ and find $\sigma^{i}(.)$.

step 2-3: Update $\widehat{W}^{i}(.)$ using (2.73). Then store $\widehat{W}^{i}(.)$ as $\breve{W}^{\zeta+1}$ and set $\zeta = \zeta + 1$.

step 2-4: If $\|\breve{W}^{\zeta} - \breve{W}^{\zeta-1}\| \geq \gamma_1$ go back to step 2-1.

step 2-5: Set $\widehat{W}_{c}^{i} = \breve{W}^{k}$, $\breve{W}^{1} = \widehat{W}_{c}^{i}$ and $\zeta = 1$.

step 3: *Calculate $\hat{v}^{i+1}(.)$ from (2.32).*

step 4: *If $\|\widehat{W}_{c}^{i-1} - \widehat{W}_{c}^{i}\| > \gamma_2$, set $i = i+1$ and go back to step 2.*

step 5: *Set $W^{*} = \widehat{W}_{c}^{i}$ and stop training.*

2.6.4. Concurrent Training

The convergence analysis of the discussed online training is fundamentally based on PE condition. In fact, the boundedness of the error signal, $\widetilde{W}^i(.)$, is the direct result of PE assumption of signal $\bar{\sigma}^i(.)$. In general, PE condition is a restrictive assumption which is difficult to be evaluated online [14, 15, 50, 53, 54, 77]. In case of the switched systems, applying PE condition becomes even more challenging. Loosely speaking, in online training of the switched systems, since samples are selected along the state trajectories, they may lack enough diversity and richness which could result in a long time training process. A remedy to solve this problem is using the so-called concurrent training method [14].

In general, concurrent training algorithms use some carefully selected data along with on-trajectory data for training and provide easy-to-check conditions for evaluating the richness of the selected data [14, 15]. Based on the stored data, one can categorize the concurrent training algorithms into two major categories. In the first category, the storage data is selected from the previous on-trajectory data which the system has already experienced [14, 76]. In the second method, the stored data is selected from the off-trajectory data and in fact uses the *simulation of experience* instead of experience in training [54]. Through simulation of experience, one does not need to drive the system, by some random inputs, to experience off-trajectory random training points. Alternatively, one can simply simulate the effect of off-trajectory data and decide whether each simulated data is helpful for the purpose of training or not. For this decision one can use the richness condition which is discussed in the sequel. The desired aspect of the second concurrent training method is that it eliminates the requirement for any probing signal to excite the system during the training process [54]. This aspect of the second method seems to be appealing to be used in the switched systems due to discussed difficulties in generating such signals for exciting the switched systems. Hence, this section studies the second concurrent training algorithm in the switched systems.

Before introducing the concurrent training law, it is important to note that the stack of off-trajectory training patterns used in this research is not necessarily selected offline. In fact the stack of off-trajectory data in this study is updated at any time the critic weights are updated. However, unlike PE condition, at each time, the richness condition for the stack of off-trajectory

data can be evaluated online. Also as mentioned before, the system does not need to be driven to experience those off-trajectory patterns and one can simply simulate their effect. The learning law with concurrent training can be denoted as [54]

$$\begin{aligned}\dot{\hat{W}}^i(t) =& -\alpha_1 \frac{\sigma^i(t)}{m_s^2(t)}\Big(Q(x)+\widehat{W}^{i^T}(t)\sigma^i(t)\Big)\\ &-\alpha_2\sum_{z=1}^{\overline{N}}\frac{\sigma_z^i}{m_{sz}^2}\Big(Q(x_z)+\widehat{W}^{i^T}(t)\sigma_z^i\Big),\end{aligned} \tag{2.74}$$

where α_1 and α_2 are constant positive learning rates. The summation term refers to the storage stack where $\overline{N}$ shows the number of stored samples. Meanwhile, x_z refers to the z^{th} sample in the storage stack where $z \in \{1,2,...,\overline{N}\}$. Also, σ_z^i and m_{sz} refer to value of $\sigma^i(.)$ and $m_s(.)$ calculated at x_z. Defining the weight error as $\widetilde{W}^i(t) = W^i - \widehat{W}^i(t)$, by noting that W^i is time invariant one can find the dynamics of the weight error as

$$\begin{aligned}\dot{\widetilde{W}}^i(t) =&\alpha_1 \frac{\sigma^i(t)}{m_s^2(t)}\Big(Q(x)+\widehat{W}^{i^T}(t)\sigma^i(t)\Big)\\ &+\alpha_2\sum_{z=1}^{\overline{N}}\frac{\sigma_z^i}{m_{sz}^2}\Big(Q(x_z)+\widehat{W}^{i^T}(t)\sigma_z^i\Big).\end{aligned} \tag{2.75}$$

Substituting $\widehat{W}^i(t) = W^i - \widetilde{W}^i(t)$, $Q(x) = \varepsilon_H^i(x) - W^{i^T}(t)\sigma^i(t)$ and $Q(x_z) = \varepsilon_{Hz}^i - W^T(t)\sigma_z^i$, one can rewrite the error dynamics as

$$\begin{aligned}\dot{\widetilde{W}}^i(t) =&\alpha_1 \bar{\sigma}^i(t)\Big(\frac{\varepsilon_H^i(x)}{m_s(t)}-\widetilde{W}^{i^T}(t)\bar{\sigma}^i(t)\Big)\\ &+\alpha_2\sum_{z=1}^{\overline{N}}\bar{\sigma}_z^i\Big(\frac{\varepsilon_{Hz}^i}{m_{sz}}-\widetilde{W}^{i^T}(t)\bar{\sigma}_z^i\Big),\end{aligned} \tag{2.76}$$

where $\bar{\sigma}^i(t) = \frac{\sigma^i(t)}{m_s(t)}$ and $\bar{\sigma}_z^i = \frac{\sigma_z^i}{m_{sz}}$. Before presenting the main results the following assumption is required to relax the PE condition which provides the richness condition for the storage stack.

Assumption 5 *There exists a finite set of points* $\Big\{x_z | z \in \{1,2,...,\overline{N}\}\Big\}$ *in the region of interest* Ω

and two positive constants $\alpha_1, \alpha_2 \in \mathbb{R}$ *such that*

$$\inf_{t>0} \left(\alpha_1 \lambda_{min}(\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t)) + \alpha_2 \lambda_{min}(\sum_{z=1}^{\overline{N}} \bar{\sigma}_z^i \bar{\sigma}_z^{i^T}) \right) > 0, \tag{2.77}$$

where α_1 *and* α_2 *are the learning rates defined in (2.74).* $\lambda_{min}(.)$ *is an operator that gives the minimum eigenvalue of its argument and* $\overline{N}$ *is the total number of data in the data stack [54].*

Compared to the PE condition, Assumption 5 is less restrictive and it can be easily evaluated online. Hence, one can present the concurrent training algorithm as Algorithm 2.5.

Algorithm 2.5 *Concurrent Training*

step 1: *Select an initial admissible policy* $\hat{v}^0(.)$*. Measure the initial state* x_0*, and select two small numbers as convergence tolerance* $\gamma_1 > 0$*,* $\gamma_2 > 0$*, and positive real constants as* α_1*, and* α_2*. Also, set* $\zeta = 1$*, and* $i = 0$*. At last, set* $\widehat{W}^0(0) = \breve{W}^1 \in \mathbb{R}^m$ *a random vector.*

step 2: *Conduct the following inner loop.*

step 2.1: Use policy $\hat{v}^i(.)$ *to operate the system and find* $\sigma^i(.)$*. Form the storage stack based on Assumption 5.*

step 2.2: Update $\widehat{W}^i(.)$ *using Eq. (2.74). Store* $\widehat{W}^i(.)$ *as* $\breve{W}^{\zeta+1}$ *and set* $\zeta = \zeta + 1$*.*

step 2.3: If $\|\breve{W}^{\zeta} - \breve{W}^{\zeta-1}\| > \gamma_1$ *go back to step 2.1.*

step 3: *Set* $\widehat{W}_c^i = \widehat{W}^i$*,* $\breve{W}^1 = \widehat{W}_c^i$ *and* $\zeta = 1$*.*

step 4: *If* $\|\widehat{W}_c^i - \widehat{W}_c^{i-1}\| > \gamma_2$ *(for* $i \geq 1$*), update the policy as* $\hat{v}^{i+1}(.)$ *from Eq. (2.32). Also, set* $i = i + 1$ *and go back to step 2.*

step 5: *Set* $W^* = \widehat{W}^i(.)$ *and stop training.*

For evaluating the performance of the concurrent training, the following definition is required.

Definition 2.3 The equilibrium point of the dynamical system of the error given by (2.76), $\widetilde{W}_{eq}(t) = \mathbf{0}$, is said to be Uniformly Ultimately Bounded (UUB) with ultimate bound $b > 0$ if for any $a > 0$ and $t_0 > 0$, there exists a positive number $N_T = N_T(a,b)$ independent of t_0, such that $\|\widetilde{W}\| \leq b$ for all $t \geq N_T + t_0$ whenever $\|\widetilde{W}(t_0)\| \leq a$ [101]. □

Theorem 2.5 *Let the PI algorithm initiate from an initial admissible policy. Also, let Assumptions 1-5 hold. Considering the gradient descent concurrent training law as in (2.74), that is, under the proposed concurrent training algorithm, the equilibrium point of the error signal* $\widetilde{W}^i(.) = \mathbf{0}$ *will be UUB.*

Proof: Let

$$L(\widetilde{W}^i(t)) = \frac{1}{2}\widetilde{W}^{i^T}(t)\widetilde{W}^i(t). \tag{2.78}$$

From (2.78), one can see that $L(0) = 0$ and $L(\widetilde{W}^i(t)) > 0$, $\forall \widetilde{W}^i(t) \neq \mathbf{0}$. Hence, one can consider $L(.)$ as a candidate Lyapunov function. Taking the time derivative of $L(.)$ in (2.78) and substituting for $\dot{\widetilde{W}}^i(t)$ from (2.76) leads to

$$\begin{aligned}\dot{L}(\widetilde{W}^i(t)) =& \alpha_1 \widetilde{W}^{i^T}(t)\bar{\sigma}^i(t)\Big(\frac{\varepsilon_H^i}{m_s(t)} - \widetilde{W}^{i^T}(t)\bar{\sigma}^i(t)\Big) \\ &+ \alpha_2 \sum_{z=1}^{\overline{N}} \widetilde{W}^{i^T}(t)\bar{\sigma}_z^i\Big(\frac{\varepsilon_{Hz}^i}{m_{sz}} - \widetilde{W}^{i^T}(t)\bar{\sigma}_z^i\Big).\end{aligned} \tag{2.79}$$

Noting that $\widetilde{W}^{i^T}(t)\bar{\sigma}^i(t)$ is a scalar, one has $\widetilde{W}^{i^T}(t)\bar{\sigma}^i(t) = \bar{\sigma}^{i^T}(t)\widetilde{W}^i(t)$. Through some algebraic manipulations, one has

$$\begin{aligned}\dot{L}(\widetilde{W}^i(t)) =& \widetilde{W}^{i^T}(t)\Big(-\alpha_1\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t) - \alpha_2\sum_{z=1}^{\overline{N}}\bar{\sigma}_z^i\bar{\sigma}_z^{i^T}\Big)\widetilde{W}^i(t) \\ &+ \widetilde{W}^{i^T}(t)\Big(\alpha_1\bar{\sigma}^i(t)\frac{\varepsilon_H^i}{m_s(t)} + \alpha_2\sum_{z=1}^{\overline{N}}\bar{\sigma}_z^i\frac{\varepsilon_{Hz}^i}{m_{sz}}\Big).\end{aligned} \tag{2.80}$$

For any symmetric matrix A, $\lambda_{min}(A)\|x\|^2 \leq x^T A x$ where $\lambda_{min}(A)$ shows the minimum eigenvalue of A. Using this property of eigenvalues, norm properties, triangle inequality, Assumption 5, and

considering the normalization terms $m_s(t)$ and m_{sz} it follows that

$$\begin{aligned}
&\dot{L}(\widetilde{W}^i(t)) \leq \widetilde{W}^{i^T}(t)\Big(-\alpha_1 \lambda_{min}(\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t)) \\
&-\alpha_2 \lambda_{min}(\sum_{z=1}^{\overline{N}} \bar{\sigma}_z^i \bar{\sigma}_z^{i^T})\Big)\widetilde{W}^i(t) \\
&+\|\widetilde{W}^i(t)\|\Big(\alpha_1\|\bar{\sigma}^i(t)\|\|\frac{\varepsilon_H^i}{m_s(t)}|+\alpha_2\sum_{z=1}^{\overline{N}}\|\bar{\sigma}_z^i\|\|\frac{\varepsilon_{Hz}^i}{m_{sz}}|\Big) \\
\leq &-\Big(\alpha_1\lambda_{min}\Big(\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t)\Big)+\alpha_2\lambda_{min}\Big(\sum_{z=1}^{\overline{N}}\bar{\sigma}_z^i\bar{\sigma}_z^{i^T}\Big)\Big)\|\widetilde{W}^i(t)\|^2 \\
&+\varepsilon_{H,max}(\alpha_1+\alpha_2 N)\|\widetilde{W}^i(t)\|. \qquad (2.81)
\end{aligned}$$

In deriving the final form of inequality (2.81), the upper bounds for $\|\bar{\sigma}(.)\| \leq 1$ and $|\frac{\varepsilon_H^i(.)}{m_s(.)}| \leq \varepsilon_{H,max}$ were used. Considering the coefficient of $\|\widetilde{W}^i(t)\|^2$ as $-\beta_3$ and the coefficient of $\|\widetilde{W}^i(t)\|$ as β_4, one has $\dot{L}(\widetilde{W}) \leq -\beta_3\|\widetilde{W}^i(t)\|^2+\beta_4\|\widetilde{W}^i(t)\|$. This inequality is a second order expression with respect to $\|\widetilde{W}^i(t)\|$ with roots at $\|\widetilde{W}^i(t)\|=0$ and $\|\widetilde{W}^i(t)\|=\frac{\beta_4}{\beta_3}$. Based on Assumption 5, one can select the learning rates α_1 and α_2 such that $\beta_3 > 0$. Note that the existence of such $\beta_3 > 0$ is guaranteed per Assumption 5. Hence, it can be seen that if $\|\widetilde{W}(t)\| > \frac{\beta_4}{\beta_3}$ then $\dot{L}(\widetilde{W}) < 0$ which develops an upper bound for $\|\widetilde{W}^i(t)\|$. By a close look at β_4, one can see that β_4 is linearly related to $\varepsilon_{H,max}$. Hence, this bound becomes smaller as the approximation precision of the critic improves. □

2.7. Simulation Results

2.7.1. Example 1: Scalar System

Consider a first order system with the CT dynamics described as

$$\dot{x} = f_1(x(t)) = -x(t), \qquad \dot{x} = f_2(x(t)) = -x^3(t). \qquad (2.82)$$

As one can see from (2.82), the system in example 1, has two stable modes $f_1(.)$ and $f_2(.)$ and at each time t, only one of them is active. For defining the performance index, function $Q(.)$ in (2.2) is

chosen as $Q(x) = x^2(t)$. Based on the defined performance index, it is easy to see that the optimal switching schedule can be directly derived from the convergence rate of the system. As shown in [40], the optimal switching schedule can be found as

$$v^*(x(t)) = \begin{cases} 1 & \text{if } |x(t)| \leq 1, \\ 2 & \text{if } |x(t)| > 1. \end{cases} \tag{2.83}$$

For this example, the value function approximation is preformed through using a linear-in-parameter neural network. The basis functions for the critic are chosen as

$$\phi(x(t)) = [x^2(t), x^4(t), x^6(t), x^8(t), x^{10}(t)]^T. \tag{2.84}$$

As one can see from (2.84), $m = 5$ which shows the number of neurons.

For offline training, 500 random training samples $x \in [-2, 2]$ were generated. To start the training process, the initial admissible policy, $\hat{v}^0(.)$, was selected as using the first mode constantly.

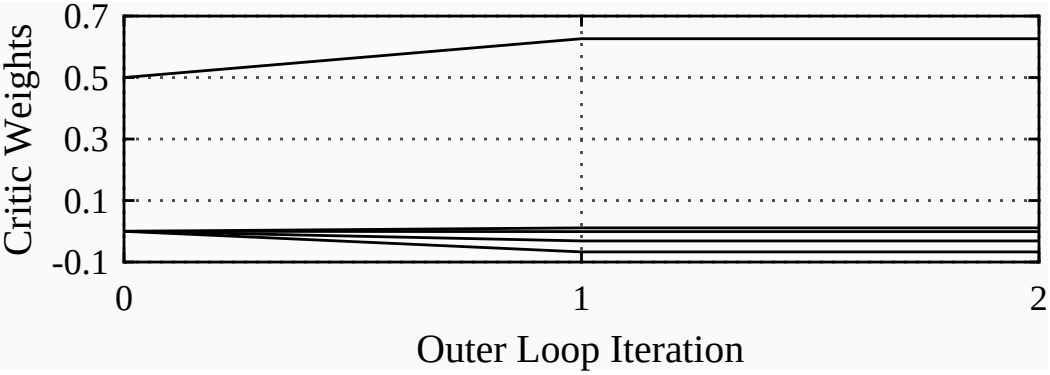

Figure 2.3: History of the critic weight parameters in offline training.

The history of the weight elements of the critic during the offline training is illustrated in Fig. 2.3. As it can be seen from Fig. 2.3, the training process converged after 3 iterations of the PI algorithm. Afterward, the trained critic was used for online scheduling with an arbitrary initial condition $x(0) = 2$.

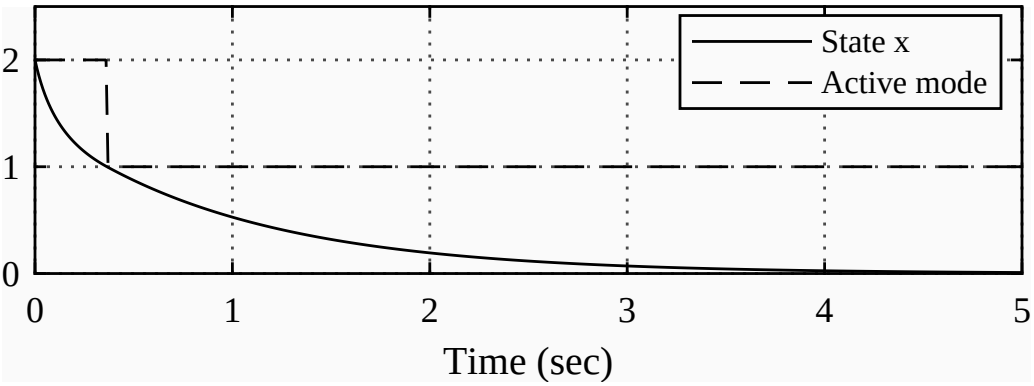

Figure 2.4: History of the state and switching schedule in online control using the controller trained offline.

The history of the state and the active modes are shown in Fig. 2.4. As one can see from Fig. 2.4, the controller switched once at the time when the respective state was passing through $x = 1$. For $|x| \leq 1$, mode 1 was active and for $|x| > 1$ mode 2 was active which demonstrates the expected optimal behavior.

2.7.2. Example 2: Second Order System

For the second example, a mass-spring-damper system is selected. Considering linear spring/damper and also neglecting friction losses, the CT dynamics can be described as (Newton's Second Law)

$$\overline{M}\ddot{x} = F(t) - kx(t) - c\dot{x}(t), \tag{2.85}$$

where $\overline{M}$, $F(.)$, k and c denote the mass of the block $(\mathbf{k}g)$ attached to a free end of a horizontal spring, external force (N), linear spring constant (N/m) and damping constant $(N.sec/m)$. Also, $x(t)$ is the horizontal displacement of the mass measured from the relaxed length of the spring (m). In this example $F(.)$ can only take discrete values, i.e., $F(t) \in \{-1, 0, 1\}$ which leads to three linear modes. We refer to the case of $F(.) = 1$ as mode 1, the case of $F(.) = -1$ as mode 2, and finally, the case of $F(.) = 0$ as mode 3. Assuming $\overline{M} = 1$ $(\mathbf{k}g)$, $k = 0.1$ (N/m), and $c = 0.1$ $(N.sec/m)$, the state-space representation of the system can be derived by taking $x_1(.)$ as the displacement from the

relaxed length of the spring and $x_2(.)$ as the velocity of mass $\overline{M}$. Hence, one has

$$\begin{aligned} \dot{x_1}(t) &= x_2(t), \\ \dot{x_2}(t) &= F(t) - 0.1x_1(t) - 0.1x_2(t). \end{aligned} \tag{2.86}$$

The control goal for the system presented in this example is bringing both states to the origin and minimizing the infinite horizon quadratic cost function defined as

$$J(x(0)) = \int_0^\infty x^T(\tau) Q_p x(\tau) d\tau, \tag{2.87}$$

where Q_p is the state penalizing matrix. In this example, Q_p is selected as a 2-by-2 identity matrix which shows the penalization of both states are enforced equally. To start the PI algorithm, one needs an admissible initial policy. In this example, using $F(.) = 0$ is an admissible policy. Also, for value function approximation, a linear-in-parameter neural network was used to implement all discussed algorithms in this chapter. The basis functions of the neurons were selected as polynomials with all possible combinations of the state variables up to the 4^{th} degree without repetitions. This choice of basis functions leads to 14 neurons which are linearly independent. Also, the region of training is selected as $\Omega = \{(x_1, x_2) || x_1| \leq 2, |x_2| \leq 2\}$ for implementing all discussed algorithms.

To start the offline training, 1000 random samples $x = [x_1, x_2]^T$ were generated in the region of training. $\gamma = 0.001$ was chosen for step 6 of Algorithm 2.1. The training terminated in 6 iterations of the PI algorithm. The history of the critic weight parameters is shown in Fig. 2.5.

For sequential offline training, random training samples in the region of training were generated to find the value functions. The learning rate was selected as $\alpha = 3$. This learning rate includes the normalization part in the learning law[1]. The history of the weights of the critic during the training process is shown in Fig. 2.6.

The online training was initiated with the same initial admissible policy as the offline training, i.e., $F(.) = 0$. For online training, learning rate was chosen as $\alpha = 10$. Also, $\delta t_1 = 0.1$ (sec) and $\delta t_2 = 0.05$ (sec) were selected. The history of the critic weights during the training process is shown

[1]Note that the normalizer in (2.73) is a constant.

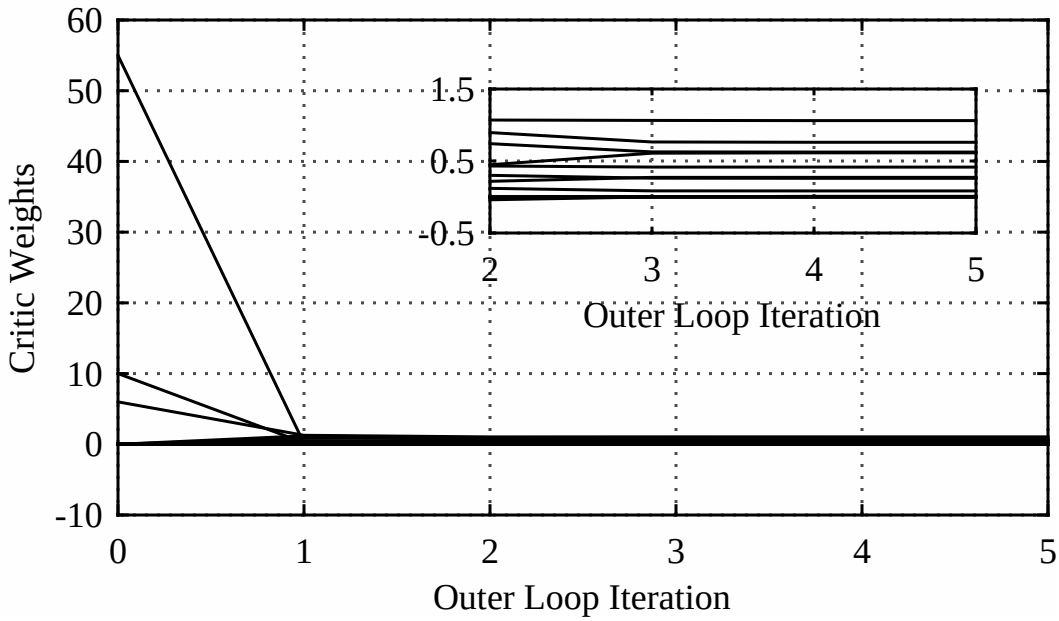

Figure 2.5: History of the critic weight parameters in batch mode offline training.

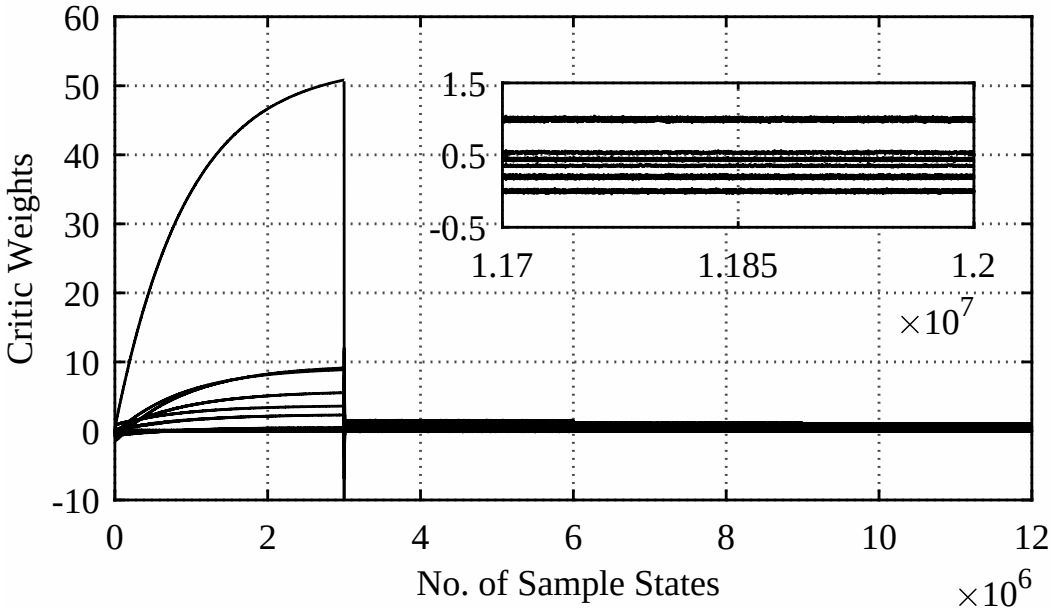

Figure 2.6: History of the critic weight parameters in sequential offline training.

in Fig. 2.7. As one can see, the training process converged in about 4.29×10^5 seconds.

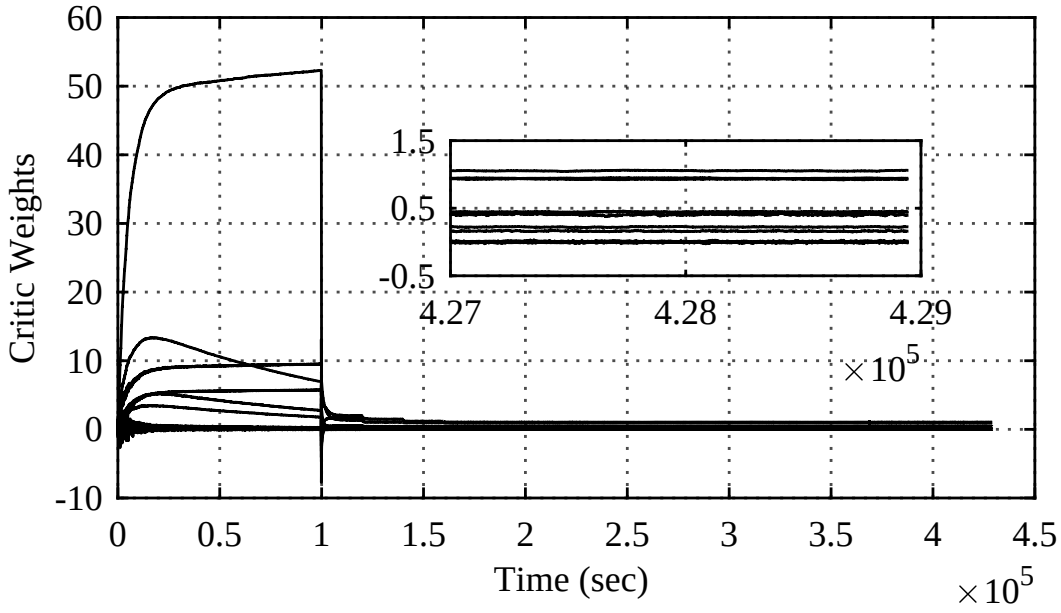

Figure 2.7: History of the critic weight parameters in online training.

For evaluating the effectiveness of the concurrent training algorithm, stacks of random states in the region of interest were used which satisfied Assumption 5. The learning rates were selected as $\alpha_1 = \alpha_2 = 300$. The training process converged in about 400 seconds. The history of the weights of the critic during training with the concurrent training algorithm is shown in Fig. 2.8.

An important feature in illustrations of online training as in Fig. 2.7 and concurrent training as in Fig. 2.8 is the significantly shorter time required for training with concurrent training method. This shows the effect of data stack on the training process. Meanwhile, the online training process uses exploring action through random switching and holding modes which adds to the length of time required for training.

To further evaluate the performance of the presented algorithms, an optimal scheduler which was introduced in [39] was simulated. This scheduler uses a VI algorithm and the training is conducted offline. For the purpose of performance evaluation, the initial condition was selected as $x(0) = [1,1]^T$ and the controllers which were trained by the presented algorithms in this chapter and [39] were used. The state trajectories are compared in Fig. 2.9. As one can see, the controllers

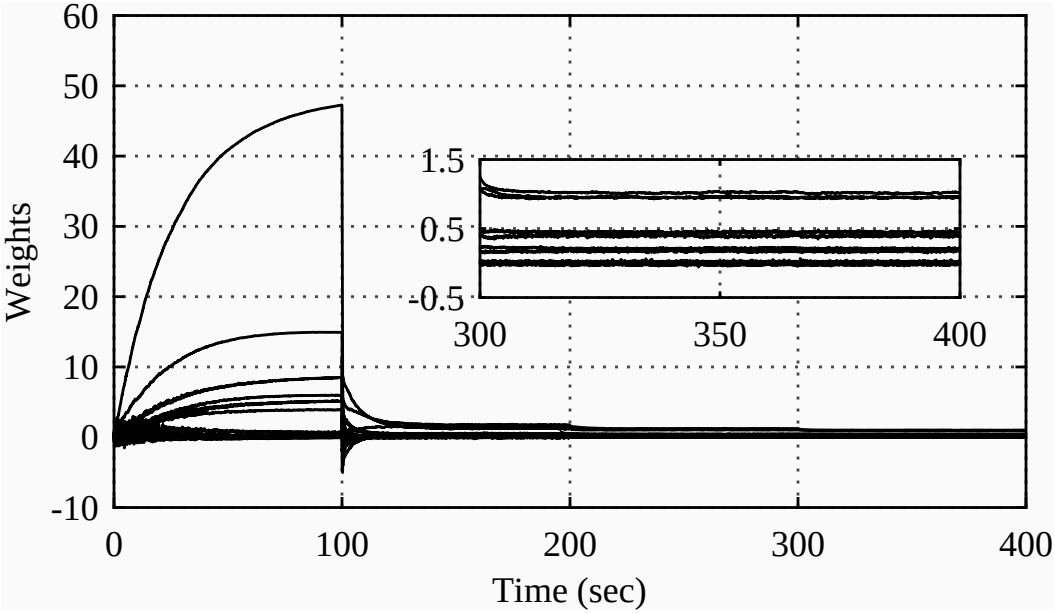

Figure 2.8: History of the critic weights parameters in concurrent training.

trained by the algorithms presented in this chapter had good and nearly identical performances in bringing the states to the origin. Also, their performance is close to that of the controller reported in [39]. For a better comparison, the errors in the distance with respect to controller used in [36] is shown in Fig. 2.10. Also, the difference between the velocity generated by controller trained in [36] and the controllers discussed in this chapter is shown in Fig. 2.11. At the end, an illustration of the switching policy is shown in Fig. 2.12.

2.8. Conclusion

Policy iteration algorithm for finding the optimal switching policy in switched systems with continuous-time dynamics and autonomous subsystems was presented. Due to the iterative nature of the policy iteration algorithm, the stability of the system governed by the evolving policies and the convergence to the optimal solution are two concerns and were discussed in details. For implementing policy iteration algorithm, offline, online, and concurrent training methods were presented. Meanwhile, a simple algorithm for designing an optimal reference tracking controller for switched systems based on offline training algorithm was presented. At the end, the performance of

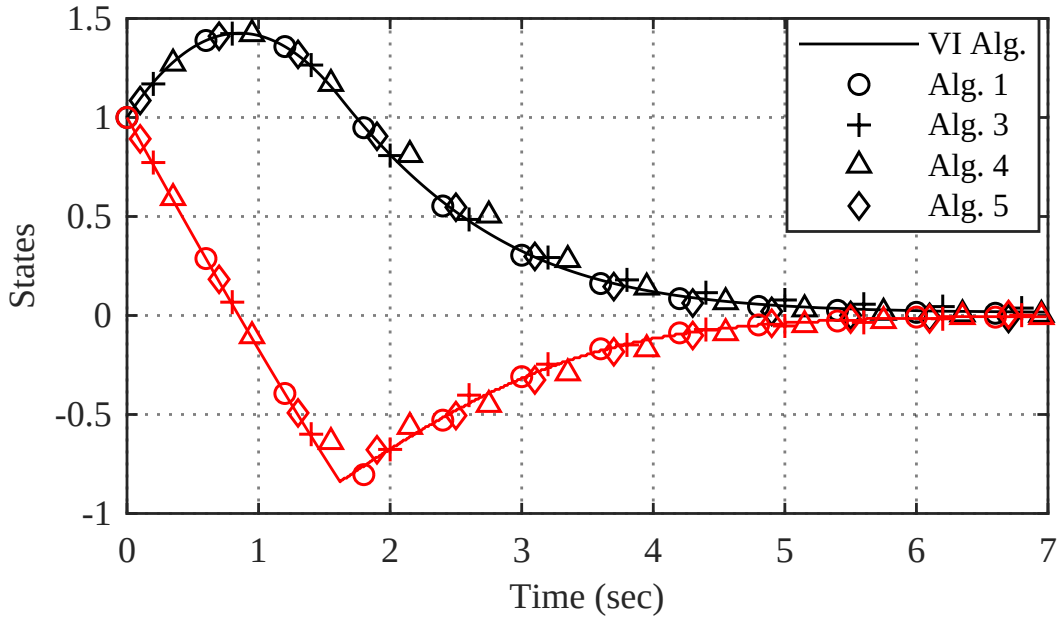

Figure 2.9: Comparison of state trajectories generated by critics trained by VI algorithm in [36], Algorithms 1, 3, 4, and 5. The initial condition is $x_0 = [1, 1]^T$. The black signals denote the distance, i.e., $x_1(.)$, and the red signals denote the velocity, i.e., $x_2(.)$.

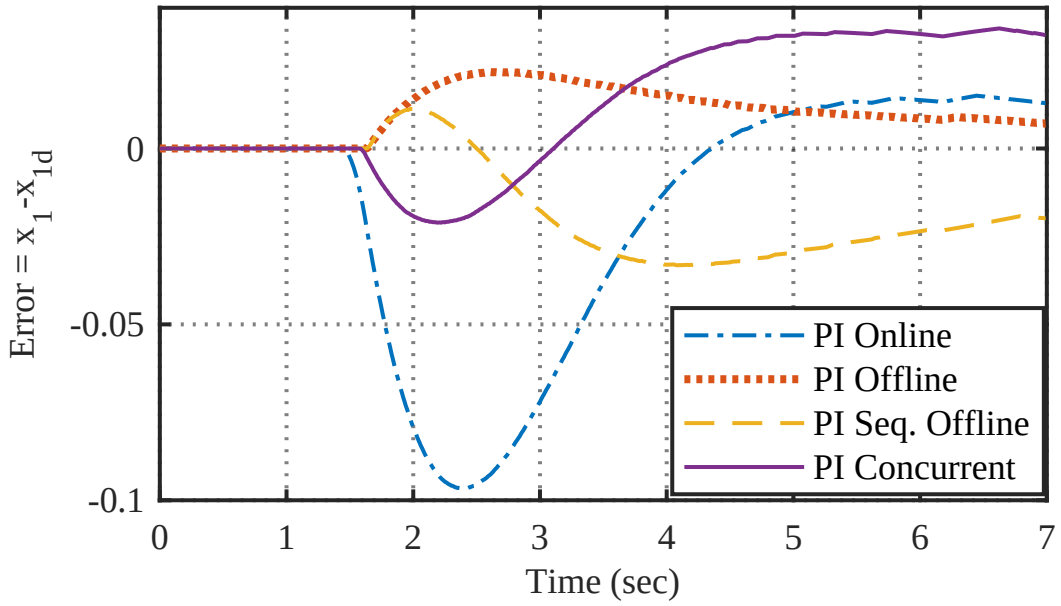

Figure 2.10: Illustration of the errors between the distance, i.e., x_1, resulted from the controller introduced in [36], and the controllers trained by Algorithms 1, 3, 4, and 5. The initial condition is $x_0 = [1, 1]^T$.

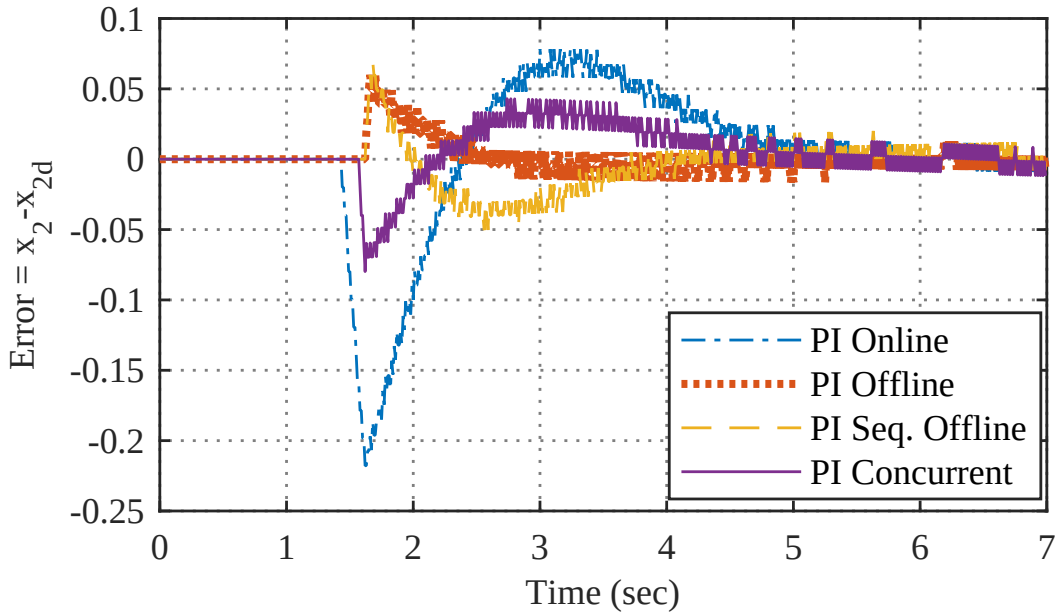

Figure 2.11: Illustration of the errors between the velocity, i.e., x_2, resulted from the controller introduced in [36], and the controllers trained by Algorithms 1, 3, 4, and 5. The initial condition is $x_0 = [1,1]^T$.

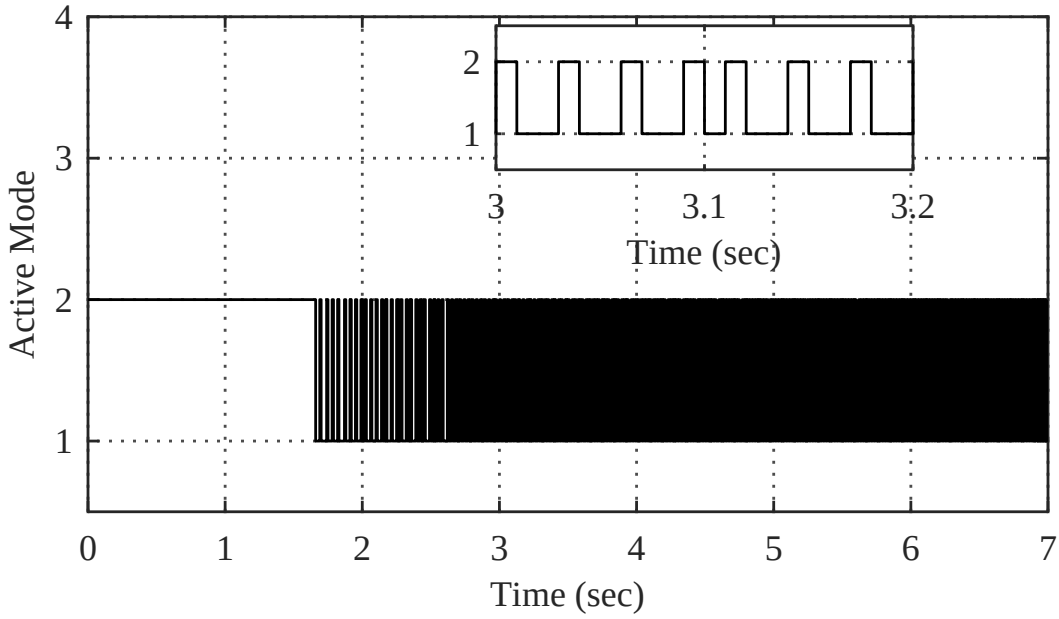

Figure 2.12: An illustration of switching policy made by the controller trained offline. The vertical axis shows the active mode where mode 1 is $F(.) = 1$, mode 2 is $F(.) = -1$ and mode 3 is $F(.) = 0$. The initial condition was $x_0 = [1,1]^T$.

the presented algorithms was compared through numerical simulations.

Chapter 3

Near-optimal Scheduling in Switched Systems with Continuous-time Dynamics: A Least Squares Approach

Two approximate solutions for optimal control of switched systems with autonomous subsystems and continuous-time dynamics are presented. The first solution formulates a policy iteration algorithm for the switched systems with recursive least squares. To reduce the computational burden imposed by the policy iteration algorithm, a second solution, called single loop policy iteration, is presented. Online and concurrent training algorithms are discussed for implementing each solution. At last, effectiveness of the presented algorithms is evaluated through numerical simulations.

Index Terms- optimal switching, approximate dynamic programming, continuous-time dynamics, least squares.

3.1. Introduction

Switched systems comprised of a finite number of autonomous subsystems (modes) with continuous-time (CT) dynamics are investigated in this researcg, where the problem is finding an optimal switching rule. At each time instant, the switching rule assigns an appropriate active mode [66, 126]. By nature, this process is a discontinuous process, called *scheduling*. In general, the discontinuous nature of scheduling and lack of smooth control signals contribute to the difficulty of controlling the switched systems [36]. Though difficult, switching control addresses many interesting engineering applications [30, 43, 49, 80, 92, 94].

In optimal control problems, the necessary and sufficient condition for optimality is given by the underlying Hamilton-Jacobi-Bellman (HJB) equation [58]. However, solving the HJB equation analytically is very difficult and in most cases impossible [58, 130]. The existing solutions for the optimal control problems such as variational calculus and dynamic programming suffer from being mathematically intractable [58] or involve curse of dimensionality [58, 62]. In order to derive an

online solution which solves the HJB equation forward-in-time and avoids the mentioned problems, Approximate Dynamic Programming (ADP) was introduced [62]. In summary, ADP typically uses neural networks to approximate cost-to-go (value function), namely *critic*, and sometimes policy, namely *actor*. ADP then uses iterative schemes to tune the unknown parameters of the neural networks to approximate the optimal solution [128, 131].

Policy Iteration (PI) algorithm is one of such iterative schemes [7]. Due to stabilizing capability of PI algorithm, this algorithm is a popular choice for online training in systems with CT dynamics [9, 20, 54, 62, 76, 78, 110, 113, 114]. The early published works such as [2, 6] studied offline training with PI algorithm. Online training with PI algorithm was reported in [113, 114] with integral reinforcement. Also, synchronous update of actor and critic in PI algorithm with gradient descent and Recursive Least Squares (RLS) training laws were studied in [110] and [9], respectively. Moreover, single online approximator was used in [20]. All mentioned PI algorithms were derived based on Persistence of Excitation (PE) condition assumption. In general, PE condition is a restrictive assumption which is difficult to verify online [76]. In order to derive a condition for the excited signal which can be checked online, the idea of concurrent training [14] in implementation of PI algorithm was introduced in [54, 76].

The reported ADP methods for the optimal switched systems can be categorized based on the representation of the dynamics of the systems [36, 39, 68, 91, 97, 109]. For systems with Discrete Time (DT) dynamics, application of Value Iteration (VI) algorithm was reported in [36, 39, 91]. In CT cases, a VI algorithm was used in [91] for control of switched systems with homogenous subsystems. In [68], a PI algorithm was used for optimal switching with controlled subsystems. The presented solution trained critic and actor neural networks and used gradient descent training laws for online training. For switched systems with autonomous subsystems and CT dynamics, off-line training was reported in [97]. For online training, [97] introduced a gradient descent training law. Another interesting approach was reported in [109] in which the idea of the synchronous PI algorithm was adapted for optimal switching. The latter paper also trained two neural networks as critic and actor where the output of the actor was a CT signal. Scheduling was performed through application of a hard limiter function which discretized the CT output of the actor.

The present study has two main contributions. First, a PI algorithm for optimal control of switched systems with autonomous subsystems and CT dynamics is formulated with an RLS training law. Meanwhile, algorithms for online and concurrent implementations of the proposed solution are presented. A major draw back of the PI algorithm is the amount of computational burden it imposes for finding the value functions of selected policies. In CT systems, this task is performed through solving a Lyapunov equation and it is called *policy evaluation*. As the second contribution of the present study, it is tried to decrease the computational burden in the PI algorithm. Hence, a new algorithm is introduced which bypasses the policy evaluation in the PI algorithm and alleviates the computational load of this algorithm. However, since the second algorithm does not solve the Lyapunov equation for policy evaluation, the evolving policies generated by this algorithm are not necessarily stabilizing. This issue necessitates existence of at least one stabilizing supervisory policy during the online training.

Compared with the recent published papers on optimal control of switched systems, the differences are as follows. Compared to [91], the present study addresses the general case of optimal switched systems with autonomous subsystems instead of switched systems with homogenous subsystems. Meanwhile, in the present study a PI algorithm is developed while in [91] the results are based on VI. Compared to [68], the structure of the system is different, i.e., the subsystems are autonomous in here while they are controlled in [68]. Meanwhile, this chapter presents RLS training laws but in [68] gradient descent training law was studied. Compared to [97], the presented RLS training laws in this chapter are derived based on an accumulated error integral. This might be the factor for significantly less time required for online training compared with the gradient descent training law based on instantaneous error presented in [97]. Moreover, the second solution and the concurrent training in this chapter are included to further remedy the heavy computational burden and lengthy training process for implementing the PI algorithm in the switched systems. Compared to [109], the second solution in this chapter is not based on synchronous PI. Also, the presented methods in this chapter explicitly solve for the optimal switching schedule hence, eliminate approximation errors of using a hard limiter function to discretize the CT output of the actor.

The rest of this chapter is organized as follows. In section 3.2, the control problem formulation is presented. The first and the second solutions are discussed in sections 3.3 and 3.4, respectively. Relaxing the PE condition and concurrent training methods are discussed in section 3.5 and simulation results are presented in section 3.6. At last, section 3.7 concludes the chapter[1].

3.2. Optimal Control Problem Formulation

In general, nonlinear dynamics of the switched systems with autonomous subsystems can be presented as

$$\dot{x}(t) = f_v(x(t)), \quad v \in \mathscr{V} = \{1,2,...,M\}, \quad x(0) = x_0 \tag{3.1}$$

where $x(.) \in \mathbb{R}^n$ is the state vector and t denotes time. The dynamics of the modes are denoted by $f_v : \mathbb{R}^n \to \mathbb{R}^n$ where subscript v identifies the active mode and $\mathscr{V}$ denotes the set of all possible modes of the system. Positive integer M is the number of modes in the system. As one can see from Eq. (3.1), at each time instant t only one mode is active.

Hereafter for notational brevity, $x = x(t)$ unless otherwise stated. The optimal control objective in this study is providing a feedback switching policy, $v(.)$, that minimizes an infinite horizon performance index (cost function) defined as

$$J(x(0)) = \int_0^\infty Q(x(\tau))d\tau \tag{3.2}$$

where $Q : \mathbb{R}^n \to \mathbb{R}$ is a positive definite function. Based on the cost function defined in Eq. (3.2), one can define cost-to-go or value function, $V : \mathbb{R}^n \to \mathbb{R}$, as the cost of going from time t to infinity as [114]

$$V(x) = \int_t^{t+\delta t} Q(x(\tau))d\tau + V(x(t+\delta t)) \tag{3.3}$$

Before formulating the optimal control problem, the following definition and assumptions are also required.

Definition 3.1 A switching policy is called admissible if it stabilizes the system presented in Eq.

[1]The preliminary results of this chapter were presented in *ASME 2016 Dynamic System and Control Conference (DSCC 2016)* [95].

(3.1) in a selected compact region of interest $\Omega \subset \mathbb{R}^n$ which includes the origin and for any initial condition $x_0 \in \Omega$, the cost-to-go at that point is finite. □

Assumption 6 *Each mode in Eq. (3.1) is Lipschitz continuous in* Ω*. Also, there exists at least one mode that* $f_v(0) = 0$*.*

Assumption 7 *There exists at least one admissible policy for the system.*

Assumption 6 helps in establishing uniqueness of resulting state trajectory in conventional systems [56]. It also guarantees that there exists some policy under which the origin is an equilibrium point for the closed loop system. As for Assumption 2, it is a controllability-like assumption which guarantees finiteness of the optimal value function.

Using Eq. (3.3) and the Bellman principle of optimality [58], one can define the optimal cost-to-go, $V^* : \mathbb{R}^n \to \mathbb{R}$, as

$$V^*(x) = \min_{v(.)} \Big(\int_t^{t+\delta t} Q(x(\tau)) d\tau + V^*(x\,(t+\delta t\,)) \Big) \tag{3.4}$$

where $x(.)$ is propagated along policy $v(.)$ in $\tau \in [t, t+\delta t]$. Assuming $\delta t \to 0$, the optimal switching policy, $v^*(.)$, can be defined as

$$v^*(x) = \arg\min_{v \in \mathscr{V}} \Big(\int_t^{t+\delta t} Q(x\,(\tau\,)) d\tau + V^*(x\,(t+\delta t\,)) \Big) \tag{3.5}$$

Equations (3.4) and (3.5) include the integral over time interval $[t, t+\delta t]$. Another approach is using the infinitesimal format of Eq. (3.4) for optimal control problem formulation. For using this approach, the following assumption is required.

Assumption 8 *The value function presented in Eq. (3.3) is continuously differentiable, i.e.,* $V(.) \in C^1$*, [36, 55, 68].*

Based on Assumption 8, one can define the infinitesimal form of Eq. (3.3) through letting $\delta t \to 0$ and using the first order approximation in Taylor series expansion of $V(x(t+\delta t))$ as

$$Q(x) + (\frac{\partial V(x)}{\partial x})^T f_{v(x)}(x) = 0 \tag{3.6}$$

where $(.)^T$ is the transpose operator. Equation (3.6) is mostly referred to as Lyapunov equation [2, 110]. In Eq. (3.3) and its infinitesimal form as in Eq. (3.6), the policy along which the state is propagated is not necessarily optimal. Considering the optimal value function $V^*(.)$ and Eq. (3.6), one can define the Hamiltonian as

$$H(x, v(.), V_x^*(.)) = Q(x) + (\frac{\partial V^*(x)}{\partial x})^T f_{v(x)}(x) \tag{3.7}$$

The minimizer of the Hamiltonian solves the HJB equation in optimal control problems [58]. Hence, one can define the HJB equation for the switched systems as

$$\begin{aligned} &\min_{v(.)} (H(x, v(.), V_x^*)) = 0 \\ &= Q(x) + (\frac{\partial V^*(x)}{\partial x})^T f_{v^*(x)}(x) \end{aligned} \tag{3.8}$$

As mentioned earlier, the HJB equation presented by Eq. (3.8) provides the necessary and sufficient condition for optimality. Noting that $Q(.)$ is a positive definite function which is not mode dependent, one can neglect its effect on minimization of the Hamiltonian. Hence, the optimal scheduler for online scheduling can be selected as

$$v^*(x) = \arg\min_{\omega \in \mathscr{V}} \Big((\frac{\partial V^*(x)}{\partial x})^T f_\omega(x) \Big) \tag{3.9}$$

In case $V^*(.)$ is known, Eq. (3.9) provides the optimal scheduling policy in a feedback form. The input of the scheduler in Eq. (3.9) is the state vector $x(.)$ and the output is an assigned mode. Hence, the computational load for feedback control calculations of this scheduler is composed of comparing M scalar valued quantities generated from substituting different modes in the term

subject to minimization on the right-hand side of Eq. (3.9) and choosing the mode which minimizes it. The challenge here is how to find the optimal value function which is the main objective of this chapter.

3.3. 1st Solution: Classic PI Algorithm

For finding $V^*(.)$, a PI algorithm is used in this section. A PI algorithm has two major steps, namely *policy evaluation* and *policy update*. In the policy evaluation step, the value function of a selected policy is sought through solving the Lyapunov equation presented in Eq. (3.6). This step is given by

$$Q(x) + (\frac{\partial V^i(x)}{\partial x})^T f_{v^i(x)}(x) = 0 \tag{3.10}$$

where the superscript i denotes the iteration index of the PI algorithm. Policy evaluation is the most computationally demanding part of the PI algorithm [62]. In policy update stage given next, the policy is updated based on the value function that is calculated in the policy evaluation step.

$$v^{i+1}(x) = \arg\min_{\omega \in \mathscr{V}} \left((\frac{\partial V^i(x)}{\partial x})^T f_\omega(x) \right) \tag{3.11}$$

An important characteristic of the PI algorithm is the stabilizing feature of the evolving policies generated by the PI algorithm in case this algorithm is initiated from a stabilizing initial policy. In summary, one starts with an initial stabilizing policy, $v^0(.)$. With $v^0(.)$ and Eq. (3.10), one finds $V^0(.)$. Then, with $V^0(.)$ and Eq. (3.11), one updates the policy and finds $v^1(.)$. The iterations continue until no meaningful differences can be detected from the value functions calculated from Eq. (3.10).

Remark 3.3.1 Stability and convergence of PI algorithm for conventional systems with control affine dynamics was presented in [5]. Motivated by [5], stability and convergence of the PI algorithm for special case of switched systems with autonomous subsystems was studied by the authors in the early results of this research given in [97]. The results presented in [97] apply to this study as well and are not repeated in this chapter. □

3.3.1. Online Implementation of PI Algorithm

In order to implement an ADP based algorithm, one can use function approximators for approximating the value functions [8, 119]. Based on Weierstrass approximation theorem, linear-in-parameter neural networks with polynomial basis functions can approximate continuous functions to any degree of precision on a compact set [93]. Also, per Assumption 8, the value function is continuous. Hence, one can define the critic for optimal value function approximation as

$$V^*(x) = W^{*T}\phi(x) + \varepsilon^*(x) \tag{3.12}$$

where $W^* \in \mathbb{R}^m$ is the optimal weight vector, $\phi : \mathbb{R}^n \to \mathbb{R}^m$ is a vector of linearly independent basis functions or polynomials and positive integer m denotes the number of neurons. $\varepsilon^* : \mathbb{R}^n \to \mathbb{R}$ denotes the error of approximating the optimal value function. Considering Eq. (3.8), substituting for $V^*(.)$ from Eq. (3.12), one has

$$Q(x) + W^{*T}\nabla\phi(x)f_{v^*(x)}(x) + \nabla^T\varepsilon^*(x)f_{v^*(x)}(x) = 0 \tag{3.13}$$

where $\nabla^T \equiv (\frac{\partial}{\partial x})^T$. Hence, one can define the residual error of the HJB equation as

$$\begin{aligned} \varepsilon_{HJB}(x) &= Q(x) + W^{*T}\nabla\phi(x)f_{v^*(x)}(x) \\ &= -\nabla^T\varepsilon^*(x)f_{v^*(x)}(x) \end{aligned} \tag{3.14}$$

Considering a selected policy $v(.)$ and Eq. (3.12), one can define the value function of the selected policy as

$$V(x) = W^T\phi(x) + \varepsilon(x) \tag{3.15}$$

where $W \in \mathbb{R}^m$ is the weight vector and $\varepsilon : \mathbb{R}^n \to \mathbb{R}$ denotes the error of approximating the value function of the selected policy, $v(.)$. Using the value function of a selected policy as in Eq. (3.15) in the Lyapunov equation presented in Eq. (3.6), one has

$$Q(x) + W^T\nabla\phi(x)f_{v(x)}(x) + \nabla^T\varepsilon(x)f_{v(x)}(x) = 0 \tag{3.16}$$

Similar to Eq. (3.14), one can define the Bellman residual error as

$$\begin{aligned} \varepsilon_H(x) &= Q(x) + W^T \nabla\phi(x) f_{v(x)}(x) \\ &= -\nabla^T \varepsilon(x) f_{v(x)}(x) \end{aligned} \tag{3.17}$$

In policy evaluation of PI, one is interested in finding W in Eq. (3.15). However, the ultimate goal of the iterative solution is finding parameters of the optimal value function, i.e., W^* in Eq. (3.12). Although equations (3.12) and (3.15) are quite similar in structure, they represent totally different targets. Equations (3.17) and (3.14) play important roles in subsequent analysis of the optimal switched systems because they directly link the effects of the approximation errors in the value function approximations to the Lyapunov and HJB equations.

In policy evaluation of PI, one needs to approximate the value function of a selected policy. Considering policy $v(.)$ as the selected policy, one has

$$\widehat{V}(x) = \widehat{W}_c^T \phi(x) \tag{3.18}$$

where $\widehat{V} : \mathbb{R}^n \to \mathbb{R}$ is the approximate value function of policy $v(.)$, and $\widehat{W}_c \in \mathbb{R}^m$ is the tunable vector of weights. In policy evaluation, one needs to find $\widehat{W}_c$ in Eq. (3.18) to approximate W in Eq. (3.15). Based on the value function approximation presented in Eq. (3.18), one can present the policy evaluation and the policy update as

$$Q(x) + \widehat{W}_c^{i^T} \nabla\phi(x) f_{v^i(x)}(x) \simeq 0 \tag{3.19}$$

$$v^{i+1}(x) = \arg\min_{\omega \in \mathscr{V}} \left(\widehat{W}_c^{i^T} \nabla\phi(x) f_\omega(x) \right) \tag{3.20}$$

Any residual value on the right-hand side of Eq. (3.19) represents the residual error with desired value as zero. In general, one needs to find $\widehat{W}_c^i$ from Eq. (3.19) at each iteration. This process, i.e., training, can be performed in different ways including, but not limited to, offline in batch mode, online with sequential data, and online with sequential data accompanied concurrently by carefully selected data, called concurrent training. In this chapter, online and concurrent training with RLS

are studied for the switched systems. For this purpose, consider the i^{th} iteration of the PI algorithm and the residual error of substituting $\widehat{V}(.)$ as in Eq. (3.18) into the Lyapunov equation as

$$e(t) = Q(x) + \widehat{W}^{i^T}(t)\nabla\phi(x)f_{v^i(x)}(x) \tag{3.21}$$

where $\widehat{W}^i(.)$ is the current estimate of $\widehat{W}_c^i$. The goal of online training with RLS algorithm is finding $\widehat{W}^i(.)$ that minimizes the integral (accumulated) cost defined as

$$\begin{aligned} J(\widehat{W}) &= \frac{1}{2}\int_0^t \frac{e^2(t)}{m_s^2(\tau)}d\tau \\ &= \frac{1}{2}\int_0^t \frac{\left(Q\left(x\left(\tau\right)\right) + \widehat{W}^{i^T}(t)\sigma^i(\tau)\right)^2}{m_s^2(\tau)}d\tau \end{aligned} \tag{3.22}$$

where $\sigma^i(t) = \nabla\phi(x)f_{v^i(x)}\left(x\right)$ and $m_s(t) = 1 + \sigma^{i^T}(t)\sigma^i(t)$ is a normalization term [45, 86] which is used for proving the convergence of the proposed algorithms.

Remark 3.3.2 The RLS derivation of PI for conventional systems with control affine dynamics was presented in [86]. For the special case of the switched systems, one needs $\left(e(t)/m_s(\tau)\right)^2$ in Eq. (3.22) to be integrable for an admissible policy in order to be able to use the RLS training laws reported in [86]. However, integrability of $\left(e(t)/m_s(\tau)\right)^2$ is not clear due to existence of discontinuities in $\sigma^i(.)$ resulted from switching. Noting that the number of switching is countable[1] [132], the integrand in Eq. (3.22) is a piecewise continuous function with countable but not necessarily finite number of simple (first kind) [93] discontinuities. The Lebesgue theorem states that discontinuous functions are integrable in Riemann sense if and only if they are bounded and the set of discontinuities forms a set of measure zero [1]. Boundedness of the integrand in Eq. (3.22) can be directly resulted from boundedness of its elements. $Q(x)$ is a continuous function and $m_s(.) \geq 1$, then $Q(x)/m_s(\tau)$ is bounded on any compact region of training since any continuous function is bounded on a compact set [93]. Meanwhile, $\sigma^i(.)/m_s(.)$ is normalized with $m_s(.) \geq 1$ so each element of $\sigma^i(.)/m_s(.)$ is less than or equal to 1 which implies boundedness. Since any countable set is a set of measure

[1] It is worthy of attention that even if 'Zeno' happens, i.e., infinite number of switching in a finite time [48, 132], the number of switching will still be countable. This countability also holds in Fuller's phenomenon in optimal control problems [11, 27] which is considered as an equivalent phenomenon to Zeno executions in hybrid systems [59, 64].

zero [93], the set of discontinuities in the integrand of Eq. (3.22) is a set of measure zero. Hence, the integrand in Eq. (3.22) satisfies the conditions stated in the Lebesgue theorem and it is integrable in Riemann sense. This result allows us to use RLS algorithm for the switched systems. □

Based on the discussion in Remark 3.3.2, one can define the RLS training laws as follows [45]

$$\dot{P}(t) = -P(t)\frac{\sigma^i(t)\sigma^{i^T}(t)}{m_s^2(t)}P(t); \quad P(0) = \alpha_0\mathbf{I} \tag{3.23}$$

$$\dot{\widehat{W}}^i(t) = -\frac{P(t)\sigma^i(t)}{m_s^2(t)}\left(Q(x) + \widehat{W}^{i^T}(t)\sigma^i(t)\right) \tag{3.24}$$

where $P(.) \in \mathbb{R}^{m\times m}$ is the covariance matrix. $\alpha_0 > 0$ is a design parameter and $\mathbf{I} \in \mathbb{R}^{m\times m}$ denotes the identity matrix. The online training with the PI algorithm is summarized in Algorithm 3.1.

Algorithm 3.1 *PI Algorithm with RLS for Switched Systems*

step 1: *Set $i = 0$ and select an initial admissible policy $v^0(.)$. Select a region of training Ω, and a random initial weight vector $\widehat{W}^0(0) \in \mathbb{R}^m$, where m is the number of neurons. Also, select a positive real number α_0 and set $P(0) = \alpha_0\mathbf{I} \in \mathbb{R}^{m\times m}$. Moreover, select two time intervals as δt_1 and δt_2, and two small positive values γ_1 and γ_2 as convergence tolerances. At last, set $\breve{W}^1 = \widehat{W}^i(0)$ and $\zeta = 1$.*

step 2: *Select a random mode let the system run with it for duration of δt_1.*

step 3: *Conduct the following inner loop:*

step 3.1: Use $v^i(.)$ to operate the system.

step 3.2: Update $P(.)$ and $\widehat{W}^i(.)$ using Eq. (3.23) and (3.24) respectively. Then store $\widehat{W}^i(.)$ as $\breve{W}^{\zeta+1}$ and set $\zeta = \zeta + 1$.

step 3.3: If $\|\breve{W}^{\zeta} - \breve{W}^{\zeta-1}\| \geq \gamma_1$, check the elapsed time from beginning of step 3. If the elapsed time is less than δt_2, go back to step 3.1. Otherwise, go back to step 2.

step 4: *Set $\widehat{W}_c^i = \widehat{W}^i(.)$. If $\|\widehat{W}_c^i - \widehat{W}_c^{i-1}\| \geq \gamma_2$ (for $i \geq 1$), update the policy, $v^{i+1}(.)$, from Eq. (3.20), set $\breve{W}^1 = \widehat{W}_c^i$, $\zeta = 1$, $P(.) = \alpha_0\mathbf{I}$, $i = i+1$ and go back to step 2. Otherwise, set $W^* = \widehat{W}_c^i$ and stop training.*

Remark 3.3.3 For establishing convergence of the PI algorithm, only convergence of the value functions in the inner loop, i.e., step 3 of Algorithm 3.1, is investigated. The convergence of the evolving value functions to the optimal value function, i.e., step 4 of Algorithm 3.1, was studied in [97]. □

3.3.2. Convergence Analysis of Algorithm 3.1

Before discussing the convergence of the value functions in step 3 of Algorithm 3.1, the following assumptions are required.

Assumption 9 *The set of basis functions and their gradients are bounded in the compact region of training* Ω*, i.e.,* $\forall x \in \Omega$, $\|\phi(x)\| \leq \phi_{max}$ *and* $\|\nabla\phi(x)\| \leq \phi_{\nabla,max}$.

Assumption 10 *The neural network approximation errors and the Bellman/HJB residual errors are bounded in the compact region of training* Ω*, i.e.,* $\forall x \in \Omega$, $|\varepsilon(x)| \leq \varepsilon_{max}$, $|\varepsilon^*(x)| \leq \varepsilon^*_{max}$, $|\varepsilon_H(x)| \leq \varepsilon_{H,max}$ *and* $|\varepsilon_{HJB}(x)| \leq \varepsilon_{HJB,max}$.

Assumption 9 can be satisfied through proper choice of basis functions [110]. Also, the boundedness of $|\varepsilon_H(.)|$ and $|\varepsilon_{HJB}(.)|$ in a compact set was investigated in [2].

Consider $\widetilde{W}^i(t) = W - \widehat{W}^i(t)$. Since W is a time-invariant vector, one can define $\dot{\widetilde{W}}^i(t) = -\dot{\widehat{W}}^i(t)$. As a result, one has

$$\dot{\widetilde{W}}^i(t) = -\dot{\widehat{W}}^i(t) = \frac{P(t)\sigma^i(t)}{m_s^2(t)}\Big(Q(x) + \widehat{W}^{iT}(t)\sigma^i(t)\Big) \tag{3.25}$$

Substituting $\widehat{W}^i(t) = W - \widetilde{W}^i(t)$, and $Q(x) + W^T\sigma^i(t) = \varepsilon_H(x)$ in Eq. (3.25), one has

$$\dot{\widetilde{W}}^i(t) = \frac{P(t)\sigma^i(t)}{m_s^2(t)}(\varepsilon_H(x) - \widetilde{W}^{iT}(t)\sigma^i(t)) \tag{3.26}$$

The convergence analysis is based on satisfaction of the PE condition which can be defined as follows [45, 110].

Definition 3.2 Signal $\sigma(t)$ is persistently excited if for all t there exist constants $\beta_1 > 0$, $\beta_2 > 0$ and $\delta t > 0$, such that

$$\beta_1 \mathbf{I} \leq \int_t^{t+\delta t} \frac{\sigma(\tau)\sigma^T(\tau)}{m_s^2(\tau)} d\tau \leq \beta_2 \mathbf{I} \tag{3.27}$$

where $\mathbf{I} \in \mathbb{R}^{m \times m}$ denotes the identity matrix [110]. □

Theorem 3.1 *Let the PI algorithm initiate from an initial admissible policy. Consider the error dynamics defined by Eq. (3.26). Let the signal $\sigma^i(.)$ be persistently excited and Assumptions 6-10 hold. Also, let $\widehat{W}^i(.)$ and $P(.)$ be updated with equations (3.24) and (3.23), respectively. Then as $t \to \infty$, an upper bound of the weight error signal $\|\widetilde{W}^i(.)\|$ converges to $\varepsilon_{H,max}\kappa\delta t$ where κ is a constant and δt is defined in Definition 3.2.*

Proof: The proof is an extended version of the one presented in [45]. Noting that $P(t)P^{-1}(t) = \mathbf{I}$ where $\mathbf{I}$ denotes the identity matrix of proper dimensions, and $\frac{d}{dt}(\mathbf{I}) = 0$, one has

$$\frac{d}{dt}(P(t)P^{-1}(t)) = \dot{P}(t)P^{-1}(t) + P(t)\frac{d}{dt}(P^{-1}(t)) = 0 \tag{3.28}$$

Substituting $\dot{P}(t)$ from Eq. (3.23) in Eq. (3.28) and multiplying both sides by $P^{-1}(t)$ leads to

$$\frac{d}{dt}(P^{-1}(t)) = \frac{\sigma^i(t)\sigma^{iT}(t)}{m_s^2(t)} \tag{3.29}$$

Now, consider the following time derivative

$$\begin{aligned} \frac{d}{dt}(P^{-1}(t)\widetilde{W}^i(t)) &= \frac{d}{dt}(P^{-1}(t))\widetilde{W}^i(t) + P^{-1}(t)\dot{\widetilde{W}}^i(t) \\ &= \frac{\sigma^i(t)\varepsilon_H(x)}{m_s^2(t)} \end{aligned} \tag{3.30}$$

In deriving Eq. (3.30), $\dot{\widetilde{W}}^i(t)$ and $\frac{d}{dt}P^{-1}(t)$ were substituted from Eq. (3.26) and Eq. (3.29), respectively. Integrating Eq. (3.30) and multiplying both sides by $P(t)$, one has

$$\widetilde{W}^i(t) = P(t)P^{-1}(0)\widetilde{W}^i(0) + P(t)\int_0^t \frac{\sigma^i(\tau)\varepsilon_H(x)}{m_s^2(\tau)} d\tau \tag{3.31}$$

Applying the norm operator on both sides of Eq. (3.31), from triangle inequality and Cauchy-Schwartz inequality one has

$$\|\widetilde{W}^i(t)\| \leq \|P(t)\|\|P^{-1}(0)\widetilde{W}^i(0)\| + \|P(t)\| \int_0^t \frac{\|\sigma^i(\tau)\||\varepsilon_H(x)|}{m_s^2(\tau)} d\tau \tag{3.32}$$

Given the normalizer $m_s(.) = 1 + {\sigma^i}^T(.)\sigma^i(.)$, one has $\|\frac{\sigma^i(.)}{m_s(.)}\| \leq 1$. Also, since $m_s(.) \geq 1$ one has $\frac{|\varepsilon_H(.)|}{m_s(.)} \leq \varepsilon_{H,max}$. Considering the above-mentioned relations one can continue as

$$\begin{aligned} \|\widetilde{W}^i(t)\| &\leq \|P(t)\|(\varepsilon_{H,max}) \int_0^t d\tau + \|P(t)\|\|P^{-1}(0)\widetilde{W}^i(0)\| \\ &\leq \|P(t)\| \Big((\varepsilon_{H,max})t + \|P^{-1}(0)\widetilde{W}^i(0)\| \Big) \end{aligned} \tag{3.33}$$

With a close look at the right-hand side of inequality (3.33), one can see that $\varepsilon_{H,max}$ and $\|P^{-1}(0)\widetilde{W}^i(0)\|$ are constants. Therefore

$$\|\widetilde{W}^i(t)\| \leq \|P(t)\|(\beta_3 t + \beta_4) \tag{3.34}$$

where $\beta_3 = \varepsilon_{H,max}$ and $\beta_4 = \|P^{-1}(0)\widetilde{W}^i(0)\|$. In [45], it was proved that PE condition results in $P(t)$ converging to zero with rate of $\frac{1}{t}$ as $t \to \infty$. More specifically, one has [45]

$$P(t) \leq \frac{\kappa}{\frac{t}{\delta t} - 1}\mathbf{I} \tag{3.35}$$

where κ is a constant and $\mathbf{I} \in \mathbb{R}^{m \times m}$ is the identity matrix. However, the decay rate of $\|P(t)\|$, used in Eq. (3.34), might not be the same as that of $P(t)$. Bringing $P(t)$ to the right-hand side of inequality (3.35), one has $\frac{\kappa}{\frac{t}{\delta t}-1}\mathbf{I} - P(t) \geq 0$. This means that each eigenvalue of $P(t)$ is less than or equal to $\frac{\kappa}{\frac{t}{\delta t}-1}$. Noting that $P(t)$ is a symmetric matrix with eigenvalues greater than or equal to zero, the singular values of $P(t)$ and hence its 2-norm coincide with their respective eigenvalues. Therefore, $\|P(t)\|_2 = \lambda_{max}(P(t)) \leq \frac{\kappa}{\frac{t}{\delta t}-1}$. This shows that $\|P(t)\|$ converges to zero with rate of $\frac{1}{t}$. Hence, one can consider $\kappa\beta_3\delta t$ as a limit upper bound of $\|\widetilde{W}^i(t)\|$ as $t \to \infty$. Since β_3 is a linear function of $\varepsilon_{H,max}$, this bound would become smaller as the approximation precision of critic improves. □

3.4. 2nd Solution: Single Loop Policy Iteration

In this section, a new algorithm, called single loop PI algorithm, is introduced. The new algorithm aims at reducing the computational load of the PI algorithm through bypassing the policy evaluation and directly searching for the optimal value function. The key idea here is using the HJB equation instead of the Lyapunov equation in formulations.

Algorithm 3.2 *Single Loop PI Algorithm*

step 1: *Select a region of training Ω, a random initial weight vector $\widehat{W}(0) \in \mathbb{R}^m$, where m is the number of neurons, and a positive real number for α_0. Initialize the covariance matrix as $P(0) = \alpha_0 \mathbf{I} \in \mathbb{R}^{m \times m}$. Also select two time intervals as δt_1 and δt_2, a fixed admissible policy and a small positive number γ_3 as a convergence tolerance. Set $\check{W}^1 = \widehat{W}(0)$, and $\zeta = 1$.*

step 2: *Select a random mode and propagate $x(.)$ along it for duration of δt_1.*

step 3: *Check for $x(.)$:*

step 3.1: If $x(.) \in \Omega$, conduct the following loop:

step 3.1.1: Operate the system using the existing policy $v(.)$.

step 3.1.2: Update $P(.)$ and $\widehat{W}(.)$ from Eq. (3.23) and (3.24), respectively. In using Eq. (3.23) and (3.24), substitute $\widehat{W}^i(.)$ with $\widehat{W}(.)$, $\sigma^i(.)$ with $\sigma(.)$ and $m_s(.)$ from Eq. (3.36). Set $\check{W}^{\zeta+1} = \widehat{W}(.)$ and $\zeta = \zeta + 1$.

step 3.1.3: If $\|\check{W}^{\zeta} - \check{W}^{\zeta-1}\| \geq \gamma_3$ check the elapsed time from starting step 3.1. If the elapsed time is less than δt_2, go back to step 3. Otherwise, go back to step 2.

step 3.2: If $x(.)$ has left Ω use the fixed stabilizing policy to bring the state vector back to the vicinity of the origin and go back to step 2.

step 4: *Set $W^* = \widehat{W}(.)$ and stop training.*

The important feature of this new algorithm is that it has only one loop in which the value function and the policy are updated. Unlike the PI algorithm, the new algorithm does not try to find the value functions of each $v^i(.)$. In fact, the new algorithm aims for finding the optimal value

function which solves the HJB equation. Hence, the approximation goal which was introduced in Eq. (3.15) is no longer required and one needs to redefine the neural network approximation goal and training error. Therefore, in this algorithm, $\widehat{V}(.)$ is the approximate optimal value function instead of the approximate value function of a selected policy. The error signal can accordingly be defined as $\overline{W}(t) = W^* - \widehat{W}(t)$. Meanwhile, unlike $\sigma^i(.)$ defined earlier for the analysis related to Algorithm 1, here the policy $v(.)$ is subject to continuous evolution, as opposed to discrete updates indexed by PI's iteration index, i. Therefore, $\sigma(.)$ is redefined as $\sigma(t) = \nabla\phi(x) f_{v(x)}(x)$ for the subsequent analysis. At the end, in order to facilitate the analysis for this algorithm, one considers the normalizing term $m_s(.)$ as

$$m_s(t) = \max_{\omega \in \mathscr{V}} \left(1 + (\nabla\phi(x) f_\omega(x))^T (\nabla\phi(x) f_\omega(x))\right) \tag{3.36}$$

The single loop PI algorithm is summarized in Algorithm 3.2. Also, the flowchart for this algorithm is illustrated in Fig. 3.1.

Remark 3.4.1 Through relaxing the policy evaluation step in the PI algorithm, Algorithm 3.2 neglects solving the Lyapunov equation which leads to generation of policies which are not necessarily stabilizing. This issue explains the existence of step 3.2 for preventing possible state divergence during the online training as a supervisory stabilizing policy. □

3.4.1. Convergence Analysis of Algorithm 3.2

For proving the convergence of Algorithm 3.2, one needs to link the training error dynamics to the HJB equation in order to address both optimality and convergence. By substituting $\widehat{W}(t) = W^* - \overline{W}(t)$ and $Q(x) = \varepsilon_{HJB}(x) - W^{*T}\nabla\phi(x) f_{v^*(x)}(x)$ derived from Eq. (3.14) into Eq. (3.24), and letting $\sigma^*(.) = \nabla\phi(x) f_{v^*(x)}(x)$ one has

$$\dot{\overline{W}}(t) = \frac{P(t)\sigma(t)}{m_s^2(t)} \left(\varepsilon_{HJB}(x) - \overline{W}^T(t)\sigma(t) + W^{*T}(\sigma(t) - \sigma^*(t))\right) \tag{3.37}$$

Theorem 3.2 *Consider the error dynamics defined by Eq. (3.37). Let the signal $\sigma(.)$ be persistently excited and Assumptions 6-10 hold. Also, let $\widehat{W}(.)$ and $P(.)$ be updated by equations (3.24) and*

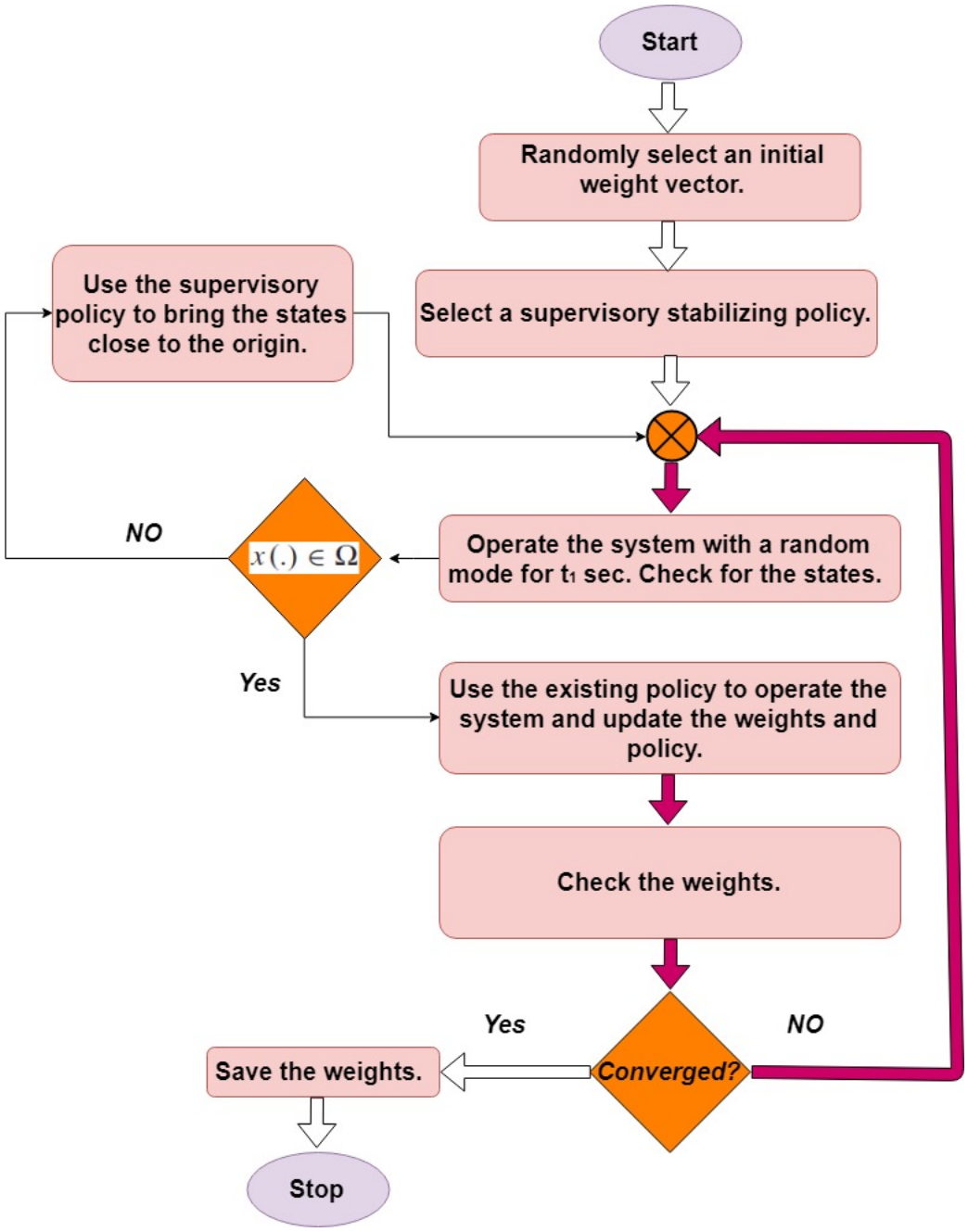

Figure 3.1: Flowchart for single loop PI algorithm.

(3.23), respectively. Then as $t \to \infty$, an upper bound of the weight error signal $\|\overline{W}(.)\|$ converges to $\kappa\delta t(\varepsilon_{HJB,max} + 2\|W^*\|)$.

Proof: With similar procedure used in the proof of Theorem 3.1, one has

$$\begin{aligned}\frac{d}{dt}(P^{-1}(t)\overline{W}(t)) &= \frac{d}{dt}(P^{-1}(t))\overline{W}(t) + P^{-1}(t)\dot{\overline{W}}(t) \\ &= \frac{\sigma(t)}{m_s^2(t)}\Big(\varepsilon_{HJB}(x) + W^{*T}(\sigma(t) - \sigma^*(t))\Big)\end{aligned} \tag{3.38}$$

In deriving Eq. (3.38), $\frac{d}{dt}P^{-1}(.)$ and $\dot{\overline{W}}(.)$ are substituted from Eq. (3.29) and (3.37), respectively. Integrating Eq. (3.38) first and then multiplying both sides by $P(t)$ one has

$$\overline{W}(t) = P(t)\left(P^{-1}(0)\overline{W}(0) + \int_0^t \frac{\sigma(\tau)}{m_s^2(\tau)}\Big(\varepsilon_{HJB}(x) + W^{*T}(\sigma(\tau) - \sigma^*(\tau))\Big)d\tau\right) \tag{3.39}$$

Applying the norm operator on both sides of Eq. (3.39), from triangle inequality and Cauchy-Schwartz inequality it follows

$$\begin{aligned}\|\overline{W}(t)\| \leq \|P(t)\|\Big(\|P^{-1}(0)\overline{W}(0)\| + \int_0^t (\|\frac{\sigma(\tau)}{m_s(\tau)}\|(\frac{|\varepsilon_{HJB}(\tau)|}{m_s(\tau)} \\ + \|\frac{W^{*T}\sigma(\tau)}{m_s(\tau)} - \frac{W^{*T}\sigma^*(\tau)}{m_s(\tau)}\|))d\tau\Big)\end{aligned} \tag{3.40}$$

Through the definition of the normalizer $m_s(.)$, one can see that $\|\frac{\sigma(.)}{m_s(.)}\| \leq 1$ and $\|\frac{\sigma^*(.)}{m_s(.)}\| \leq 1$. Hence, it follows that $\|\frac{W^{*T}\sigma(\tau)}{m_s(\tau)} - \frac{W^{*T}\sigma^*(\tau)}{m_s(\tau)}\| \leq 2\|W^*\|$. Also, per Assumption 10, $\frac{|\varepsilon_{HJB}(.)|}{m_s(.)} \leq \varepsilon_{HJB,max}$. As a result, one has

$$\|\overline{W}(t)\| \leq \|P(t)\|\left(\|P^{-1}(0)\overline{W}(0)\| + \int_0^t (\varepsilon_{HJB,max} + 2\|W^*\|)d\tau\right) \tag{3.41}$$

With the same procedure used in the proof of Theorem 1, one can see that $\varepsilon_{HJB,max}$, $\|W^*\|$ and $\|P^{-1}(0)\overline{W}(0)\|$ are constants in inequality (3.41). Therefore,

$$\|\overline{W}(t)\| \leq \|P(t)\|(\beta_5 t + \beta_6) \tag{3.42}$$

where $\beta_5 = \varepsilon_{HJB,max} + 2\|W^*\|$ and $\beta_6 = \|P^{-1}(0)\overline{W}(0)\|$. Inequality (3.42) has the same structure as inequality (3.34). Hence, using a similar argument as in that proof, due to PE condition, $\|P(t)\| \to 0$ with rate of $\frac{\kappa}{\frac{t}{\delta t}-1}$ as $t \to \infty$. Therefore, an upper bound of $\|\overline{W}(t)\|$ converges to $\kappa\beta_5\delta t$. □

3.5. Concurrent Training and Relaxing PE Condition

The convergence analysis in the discussed algorithms were based on satisfaction of the PE condition. This condition is known to be a restrictive assumption which is difficult to verify online [14, 54, 76]. The general practice for satisfying PE condition is adding probing noise signals to the inputs of system during online training [9, 20, 110]. In the case of the switched systems, such an excitation is conducted through using random switching among modes as discussed in Algorithms 3.1 and 3.2. This task results in generation of the training samples which might lack sufficient diversity and richness and could be the factor contributing to the lengthy training process. One remedy for solving this problem is through application of carefully selected data along with online data which is known as concurrent training [14,54,76]. In fact, concurrent training algorithms improve the performance of online training through application of a stack of carefully selected data which satisfies some defined richness conditions [14]. The richness conditions for eligibility of selected data to be used concurrently can be presented through simple norm conditions which can be easily evaluated online [14].

In general, one can categorize the reported concurrent training algorithms in two major categories. In the first category, the stack of stored data is selected from the on-trajectory data which system has already experienced [14, 76]. In the second category, the stack of stored data is selected randomly from the region of interest which preforms the *simulation of experience* instead of experiencing [54]. The important aspect of the latter is elimination of the requirement to excite the system with a probing noise signal during the training process. This feature of the second method is desirable in training of the switched systems. Hence, the second concurrent training algorithm is adapted in this section.

3.5.1. Concurrent Training in PI Algorithm

Along with the covariance update law as Eq. (3.23) and motivated by [54], one can define the concurrent training law in the PI algorithm as follows.

$$\dot{\widehat{W}}^i(t) = -\alpha_1 \frac{P(t)\sigma^i(t)}{m_s^2(t)}\Big(Q(x)+\widehat{W}^{i^T}(t)\sigma^i(t)\Big) - \alpha_2 \sum_{z=1}^{N} \frac{P(t)\sigma_z^i}{m_{sz}^2}\Big(Q(x_z)+\widehat{W}^{i^T}(t)\sigma_z^i\Big) \tag{3.43}$$

where z represents the summation index in the data stack formed based on a richness condition, to be discussed later. N is the total number of data in the stack. σ_z^i and m_{sz} are the values of $\sigma^i(.)$ and $m_s(.)$ calculated at x_z, i.e., the z^{th} state vector in the stack. Also, α_1 and α_2 are real positive constants. Letting $\widetilde{W}^i(t) = W - \widehat{W}^i(t)$, from the concurrent training law presented in Eq. (3.43), one can define the dynamics of the error signal as $\dot{\widetilde{W}}^i(t) = -\dot{\widehat{W}}^i(t)$. Substituting $Q(x) = \varepsilon_H(x) - W^T\sigma^i(t)$, $Q(x_z) = \varepsilon_{H_z} - W^T\sigma_z^i$ and $\widehat{W}^i(t) = W - \widetilde{W}^i(t)$ in Eq. (3.43), one has

$$\dot{\widetilde{W}}^i(t) = \alpha_1 \frac{P(t)\sigma^i(t)}{m_s^2(t)}(\varepsilon_H(x) - \widetilde{W}^{i^T}(t)\sigma^i(t)) + \alpha_2 \sum_{z=1}^{N} \frac{P(t)\sigma_z^i}{m_{sz}^2}(\varepsilon_{Hz} - \widetilde{W}^{i^T}(t)\sigma_z^i) \tag{3.44}$$

Before presenting the analysis for concurrent training, the following definition and assumption are required.

Definition 3.3 The equilibrium point $\widetilde{W}_{eq} = \mathbf{0}$ of the error dynamics given by Eq. (3.44) is called Uniformly Ultimately Bounded (UUB) with ultimate bound $b > 0$ if for any $a > 0$ and $t_0 > 0$, there exists a positive number $N_T = N_T(a,b)$ independent of t_0, such that $\|\widetilde{W}(t)\| \leq b$ for all $t \geq N_T + t_0$ whenever $\|\widetilde{W}(t_0)\| \leq a$ [110]. □

Assumption 11 *There exists a set of points* $\Big\{x_z | z \in \{1,2,...,N\}\Big\}$ *and two positive constants* $\alpha_1, \alpha_2 \in \mathbb{R}$ *such that*

$$\inf_{t\geq 0}\Big((\alpha_1 - \frac{1}{2})\lambda_{min}(\frac{\sigma^i(t)\sigma^{i^T}(t)}{m_s^2(t)}) + \alpha_2\lambda_{min}(\sum_{z=1}^{N}\frac{\sigma_z^i\sigma_z^{i^T}}{m_{sz}^2})\Big) > 0 \tag{3.45}$$

where $\lambda_{min}(.)$ *is the minimum eigenvalue of the argument.*

Assumption 11 is motivated by [54] and provides the condition for selecting the data stack to be used concurrently along the on-trajectory data.

Remark 3.5.1 In concurrent training algorithms such as [54], one forms the data stack offline. In this chapter the concept of simulation of experience [54] is used to form the data stack which is not necessarily offline. This process, unlike the PE condition, has an easy to check richness condition which can be evaluated online. Meanwhile, unlike online training that system needs to be operated to experience each training pattern, in this form of concurrent training one does not need to drive the system to experience different situations. Alternatively, one can study the effect of each off-trajectory data through simulation to see whether that data is useful for the training purpose or not. □

Remark 3.5.2 Application of a concurrent training method for adaptive model reference control in switched systems with controlled subsystems is recently reported in [29]. In [29], the switching instants are known a priori and the concurrent training is used to identify the dynamics of subsystems. Another application is reported in [51] where an algorithm is presented to form the data stacks in a concurrent training method and the updates of the stacks are considered as a switching behavior. Compared with [29] and [51], in the present study the concurrent training method is used for value function approximation and not system identification. It is worthy of attention that value function approximation is possible and data stacks can be constructed as long as the switching policy is known. In the present study, the policy is known and therefore data stacks can be constructed. □

Concurrent training in the standard PI algorithm for the switched systems is summarized in Algorithm 3.3 and the convergence of this algorithm is discussed in Theorem 3.3.

Algorithm 3.3 *Concurrent Training in PI Algorithm*

step 1: *Select an initial admissible policy $v^0(.)$, an initial state x_0, two small numbers as convergence tolerance $\gamma_1 > 0$, $\gamma_2 > 0$, and positive real constants as α_0, α_1, and α_2. Also, set $P(0) = \alpha_0 \mathbf{I} \in \mathbb{R}^{m\times m}$, $\zeta = 1$, and $i = 0$. At last, set $\widehat{W}^0(0) = \check{W}^1 \in \mathbb{R}^m$ a random vector.*

step 2: *Conduct the following inner loop.*

step 2.1: Use policy $v^i(.)$ to operate the system and find $\sigma^i(.)$. Form the storage stack based on Assumption 11.

step 2.2: *Update $P(.)$ from Eq. (3.23) and $\widehat{W}^i(.)$ using Eq. (3.43). Store $\widehat{W}^i(.)$ as $\breve{W}^{\zeta+1}$ and set* $\zeta = \zeta + 1$.

step 2.3: *If $\|\breve{W}^{\zeta} - \breve{W}^{\zeta-1}\| > \gamma_1$ go back to step 2.1.*

step 3: *Set $\widehat{W}_c^i = \widehat{W}^i$, $\breve{W}^1 = \widehat{W}_c^i$ and $\zeta = 1$.*

step 4: *If $\|\widehat{W}_c^i - \widehat{W}_c^{i-1}\| > \gamma_2$ (for $i \geq 1$), update the policy as $v^{i+1}(.)$ from Eq. (3.20). Also, set $P(.) = \alpha_0 \mathbf{I}$, $i = i+1$ and go back to step 2.*

step 5: *Set $W^* = \widehat{W}^i$ and stop training.*

Theorem 3.3 *Let the PI algorithm be initiated from an initial admissible policy. Also, let Assumptions 6-11 hold. Then through the concurrent update law presented in Eq. (3.43) along with the covariance matrix update law presented in Eq. (3.23), the equilibrium point of the error signal $\widetilde{W}^i(.) = \mathbf{0}$ will be UUB.*

Proof: Consider the following candidate Lyapunov function

$$L(t) = \frac{1}{2}\widetilde{W}^{i^T}(t)P^{-1}(t)\widetilde{W}^i(t) \tag{3.46}$$

Taking the time derivative of $L(t)$ in Eq. (3.46) and substituting for $\dot{\widetilde{W}}^i(t)$ from Eq. (3.44), and $\frac{d}{dt}(P^{-1}(t))$ from Eq. (3.29) leads to

$$\begin{aligned}\dot{L}(t) = &-\widetilde{W}^{i^T}(t)\Big((\alpha_1 - \frac{1}{2})\bar{\sigma}^i(t)\bar{\sigma}^{i^T}(t) + \alpha_2 \sum_{z=1}^{N} \bar{\sigma}^i{}_z \bar{\sigma}^{i^T}_z\Big)\widetilde{W}^i(t) \\ &+ \widetilde{W}^{i^T}(t)\Big(\alpha_1 \bar{\sigma}^i(t)\frac{\varepsilon_H(x)}{m_s(t)} + \alpha_2 \sum_{z=1}^{N} \bar{\sigma}^i{}_z \frac{\varepsilon_{Hz}}{m_{sz}}\Big)\end{aligned} \tag{3.47}$$

where $\bar{\sigma}^i(.) = \frac{\sigma^i(.)}{m_s(.)}$ for notational brevity. Considering any arbitrary symmetric matrix A, one has $\lambda_{min}(A)\|x\|^2 \leq x^T A x \leq \lambda_{max}(A)\|x\|^2$. Considering Assumptions 1-11, using the aforementioned

inequality, norm properties and triangle inequality, it is straight forward to show that

$$\dot{L}(t) \leq -\Big((\alpha_1 - \frac{1}{2})\lambda_{min}(\bar{\sigma}^i(t)\bar{\sigma}^{iT}(t)) + \alpha_2\lambda_{min}(\sum_{z=1}^{N}\bar{\sigma}^i{}_z\bar{\sigma}^{iT}_z)\Big)$$
$$* \|\widetilde{W}^i(t)\|^2 + (\alpha_1 + N\alpha_2)\varepsilon_{H,max}\|\widetilde{W}^i(t)\| \tag{3.48}$$

Denoting the coefficient of $\|\widetilde{W}^i(t)\|^2$ in Eq. (3.48) by $-\beta_7$ and defining $\beta_8 = (\alpha_1 + N\alpha_2)\varepsilon_{H,max}$, one has $\dot{L}(t) \leq -\beta_7\|\widetilde{W}^i(t)\|^2 + \beta_8\|\widetilde{W}^i(t)\|$. α_1 and α_2 are design parameters and can be selected such that the coefficient of $\|\widetilde{W}^i(t)\|^2$ is negative, i.e., $\beta_7 > 0$. The existence of such $\beta_7 > 0$ is guaranteed per Assumption 11. Hence, one can see that if $\|\widetilde{W}^i(t)\| > \frac{\beta_8}{\beta_7}$, $\dot{L}(t)$ becomes negative which leads to an upper bound for the weight error. Considering β_8, one can see that it is linearly related to the magnitude of approximation error. Therefore, the (ultimate) upper bound becomes smaller as the approximation capability improves. □

3.5.2. Concurrent Training in Single Loop PI Algorithm

In this section, the effect of concurrent training on the single loop PI algorithm is discussed. Consider the normalizing term, $m_s(.)$, as Eq. (3.36), the covariance update law as Eq. (3.23) and the weight update law as Eq. (3.43)[1]. Also, consider the approximation goal of the neural network as the optimal value function instead of the value function of a selected policy. Therefore one has $\overline{W}(t) = W^* - \widehat{W}(t)$ and $\dot{\overline{W}}(t) = -\dot{\widehat{W}}(t)$. By substituting $Q(x) = \varepsilon_{HJB}(x) - W^{*T}\sigma^*(t)$, $Q(x_z) = \varepsilon_{HJB_z} - W^{*T}\sigma^*_z$, and $\widehat{W}(t) = W^* - \overline{W}(t)$, one has

$$\dot{\overline{W}}(t) = \alpha_1\frac{P(t)\sigma(t)}{m_s^2(t)}\Big(\varepsilon_{HJB}(x) + W^{*T}(\sigma(t) - \sigma^*(t)) - \overline{W}^T(t)\sigma(t)\Big)$$
$$+ \alpha_2\sum_{z=1}^{N}\frac{P(t)\sigma_z}{m_{s_z}^2}\Big(\varepsilon_{HJB_z} + W^{*T}(\sigma_z - \sigma^*_z) - \overline{W}^T(t)\sigma_z\Big) \tag{3.49}$$

Theorem 3.4 *Consider the single loop PI algorithm. Let Assumptions 6-11 hold. Under the*

[1]Similar to presentation of Algorithm 2, in using the above mentioned equations one substitutes $\sigma^i(.)$ and $\widehat{W}^i(.)$ by $\sigma(.)$ and $\widehat{W}(.)$, respectively.

concurrent update law presented in Eq. (3.43) along with the covariance matrix update presented in Eq. (3.23), the equilibrium point of the error signal $\overline{W}(.) = \mathbf{0}$ *will be UUB.*

Proof: Consider the following candidate Lyapunov function.

$$L(t) = \frac{1}{2}\overline{W}^T(t)P^{-1}(t)\overline{W}(t) \tag{3.50}$$

Taking the time derivative of $L(t)$ in Eq. (3.50) and then substitute for $\frac{d}{dt}P^{-1}(t)$ and $\dot{\overline{W}}(t)$ form equations (3.29) and (3.49), one has

$$\begin{aligned}\dot{L}(t) = &-\overline{W}^T(t)\Big((\alpha_1 - \frac{1}{2})\bar{\sigma}(t)\bar{\sigma}^T(t) + \alpha_2\sum_{z=1}^{N}\bar{\sigma}_z\bar{\sigma}_z{}^T\Big)\overline{W}(t)\\ &+\overline{W}^T(t)\Bigg(\alpha_1\bar{\sigma}(t)\frac{\varepsilon_{HJB}(x)}{m_s(t)} + \alpha_1\bar{\sigma}(t)W^{*T}(\bar{\sigma}(t) - \bar{\sigma}^*(t))\\ &+\alpha_2\sum_{z=1}^{N}\Big(\bar{\sigma}_z\frac{\varepsilon_{HJB_z}}{m_{sz}} + \bar{\sigma}_z W^{*T}(\bar{\sigma}_z - \bar{\sigma}_z^*)\Big)\Bigg)\end{aligned} \tag{3.51}$$

where $\bar{\sigma}(.) = \frac{\sigma(.)}{m_s(.)}$ for notational brevity. Applying norm operator on the result, by using eigenvalue properties for a symmetric matrix A such that $\lambda_{min}(A)\|x\|^2 \leq x^T Ax$, triangle and Cauchy-Schwartz inequalities one can develop the following inequality

$$\begin{aligned}\dot{L}(t) \leq &-\Big((\alpha_1 - \frac{1}{2})\lambda_{min}(\bar{\sigma}(t)\bar{\sigma}^T(t)) + \alpha_2\lambda_{min}(\sum_{z=1}^{N}\bar{\sigma}_z\bar{\sigma}_z{}^T)\Big)\\ &*\|\overline{W}(t)\|^2 + \Bigg(\alpha_1\|\bar{\sigma}(t)\|\frac{|\varepsilon_{HJB}(x)|}{m_s(t)} + \alpha_1\|\bar{\sigma}(t)\|\|W^*\|\\ &*\|\bar{\sigma}(t) - \bar{\sigma}^*(t)\| + \alpha_2\sum_{z=1}^{N}\|\bar{\sigma}_z\|\frac{|\varepsilon_{HJB_z}|}{m_{sz}}\\ &+\alpha_2\sum_{z=1}^{N}(\|\bar{\sigma}_z\|\|W^*\| * (\|\bar{\sigma}_z - \bar{\sigma}_z^*\|)\Bigg)\|\overline{W}(t)\|\end{aligned} \tag{3.52}$$

Similar to the proof of Theorem 3.3, by using the upper bounds for each term in inequality (3.52) one has

$$\dot{L}(t) \leq -\Big((\alpha_1 - \frac{1}{2})\lambda_{min}(\bar{\sigma}(t)\bar{\sigma}^T(t)) + \alpha_2\lambda_{min}(\sum_{z=1}^{N}\bar{\sigma}_z\bar{\sigma}_z^T)\Big)$$
$$* \|\overline{W}(t)\|^2 + (\varepsilon_{HJB,max} + 2\|W^*\|)(\alpha_1 + \alpha_2 N)\|\overline{W}(t)\| \tag{3.53}$$

Considering $-\beta_9$ and β_{10} as the coefficients of $\|\overline{W}(t)\|^2$ and $\|\overline{W}(t)\|$ in Eq. (3.53), one can see that $\dot{L}(t) \leq -\beta_9\|\overline{W}(t)\|^2 + \beta_{10}\|\overline{W}(t)\|$. In case α_1 and α_2 are selected such that $\beta_9 > 0$ then $\|\overline{W}(t)\| > \frac{\beta_{10}}{\beta_9}$ leads to $\dot{L}(t)$ being negative. Similar to the proof of Theorem 3.3, existence of such a positive β_9 is guaranteed per Assumption 11. □

The concurrent training with the single loop PI algorithm for the switched systems is presented in Algorithm 3.4.

Algorithm 3.4 *Concurrent Training in Single Loop PI Algorithm*

step 1: *Select a region of training Ω and a supervisory admissible policy. Also, select an initial state x_0, a small positive real number as convergence tolerance γ_3, and positive real numbers as α_0, α_1, and α_2. Moreover, set $\zeta = 1$ and $\breve{W}^1 = \widehat{W}(0) \in \mathbb{R}^m$ a random vector where m is the number of neurons. At last, initialize the covariance matrix $P(0) = \alpha_0\mathbf{I} \in \mathbb{R}^{m\times m}$.*

step 2: *Check $x(.)$:*

step 2.1: If $x(.) \in \Omega$:

step 2.1.1: Use policy $v(.)$ and find $\sigma(.)$ and $m_s(.)$ from Eq. (3.36).

step 2.1.2: Form the storage stack based on Assumption 11.

step 2.1.3: Update $P(.)$ and $\widehat{W}(.)$ from Eqs. (3.23) and (3.43) by using $\sigma(.)$ and $\widehat{W}(.)$ instead of $\sigma^i(.)$ and $\widehat{W}^i(.)$. Store $\widehat{W}(.)$ as $\breve{W}^{\zeta+1}$.

step 2.1.4: Update $v(.)$ from Eq. (3.20) by using $\widehat{W}(.)$ instead of $\widehat{W}^i(.)$. Operate the system using this policy.

step 2.2: If $x(.) \notin \Omega$:

step 2.2.1: Use the supervisory control and bring the states close to the origin. Then go to step 2.1.

step 3: *If* $\|\widehat{W}^{\zeta+1} - \widehat{W}^{\zeta}\| > \gamma_3$, *set* $\zeta = \zeta + 1$ *and go back to step 2.*

step 4: *Set* $W^* = \widehat{W}(.)$ *and stop training.*

3.6. Numerical Simulation

In order to evaluate the effectiveness of the presented algorithms, a linear second order system is selected. Considering a mass-spring-damper system, the CT dynamics of this system can be presented as

$$\overline{M}\ddot{x} = F(t) - kx(t) - c\dot{x}(t) \tag{3.54}$$

where $\overline{M}$ denotes mass of a block $(\mathbf{k}g)$ attached to the end of a horizontal spring, $F(t)$ denotes the external force acting on the mass (N), k is the linear spring constant (N/m), $x(t)$ is the displacement of the mass measured from the relaxed length of the spring (m), and c is the damping constant $(N.sec/m)$. Assuming $\overline{M} = 1$ $(\mathbf{k}g)$, $k = 0.1$ (N/m) and $c = 0.1$ $(N.sec/m)$, the state space representation of this system can be derived by selecting $x_1(t) = x(t)$ and $x_2(t) = \dot{x}(t)$. Hence, one has

$$\begin{aligned} \dot{x_1}(t) &= x_2(t) \\ \dot{x_2}(t) &= F(t) - 0.1x_1(t) - 0.1x_2(t) \end{aligned} \tag{3.55}$$

In order to generate a switching behavior in the system, it is assumed that $F(t)$ can only have 3 discrete amounts as $F(t) \in \{-1, 0, 1\}$. Based on the presented discrete $F(t)$ one has three different modes. We refer to the case of $F(.) = 1$ as mode 1, the case of $F(.) = -1$ as mode 2, and finally, the case of $F(.) = 0$ as mode 3. The control goal in this example is bringing both states to the origin by minimizing a quadratic cost function defined as

$$J(x(0)) = \int_0^{\infty} x^T(\tau) Q_p x(\tau) d\tau \tag{3.56}$$

where Q_P is a positive definite state penalizing matrix selected as $Q_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. In this example, no force policy, i.e., $F(t) = 0$, is a stabilizing policy which can be used as required in the discussed algorithms. For value function approximation, a linear-in-parameter neural network with basis functions comprised of polynomials with all possible combinations of the state variables up to the 4^{th} degree without repetitions is selected. This choice of basis functions leads to 14 neurons which are linearly independent. Also, the region of training is confined to $\Omega = \{(x_1, x_2) | |x_1| \leq 2, |x_2| \leq 2\}$ for implementing all discussed algorithms.

For implementing Algorithm 3.1, α_0 was selected as 600. Also, $\delta t_1 = 0.3$ (sec) and $\delta t_2 = 0.05$ (sec) were selected. The initial critic weight elements were selected as a random vector and γ_1 and γ_2 were selected as 1×10^{-12} and 0.05, respectively. The training process converged after 6 iterations of the PI algorithm in about 6.2×10^4 seconds. The history of critic weight elements in online training with the PI algorithm is shown in Fig. 3.2.

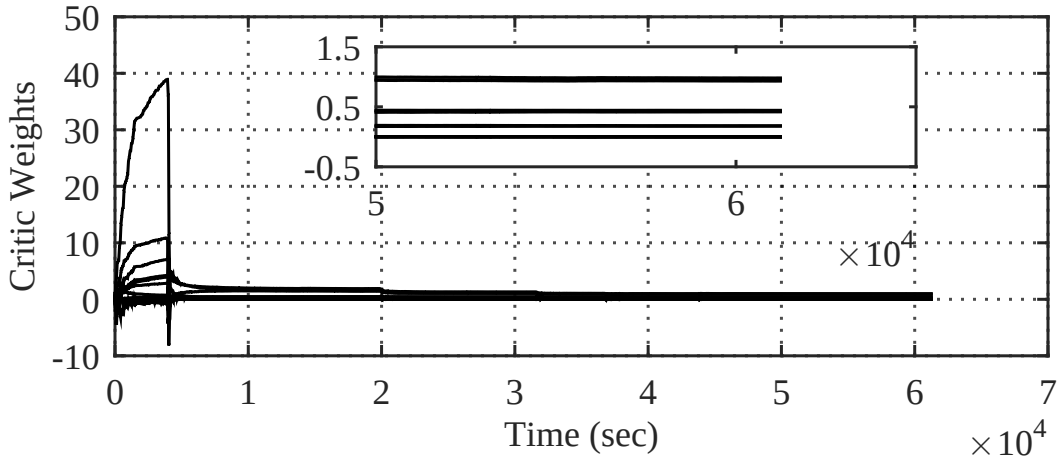

Figure 3.2: History of the critic weights in online training by Algorithm 3.1.

For implementing Algorithm 3.2, $\gamma_3 = 1 \times 10^{-10}$, $\alpha_0 = 350$, $\delta t_1 = 0.5$ (sec) and $\delta t_2 = 0.05$ (sec) were selected. The history of the critic weight elements in online training with Algorithm 3.2 is shown in Fig. 3.3. As one can see from Fig. 3.3, the training process converged much faster, in about 2.2×10^4 seconds.

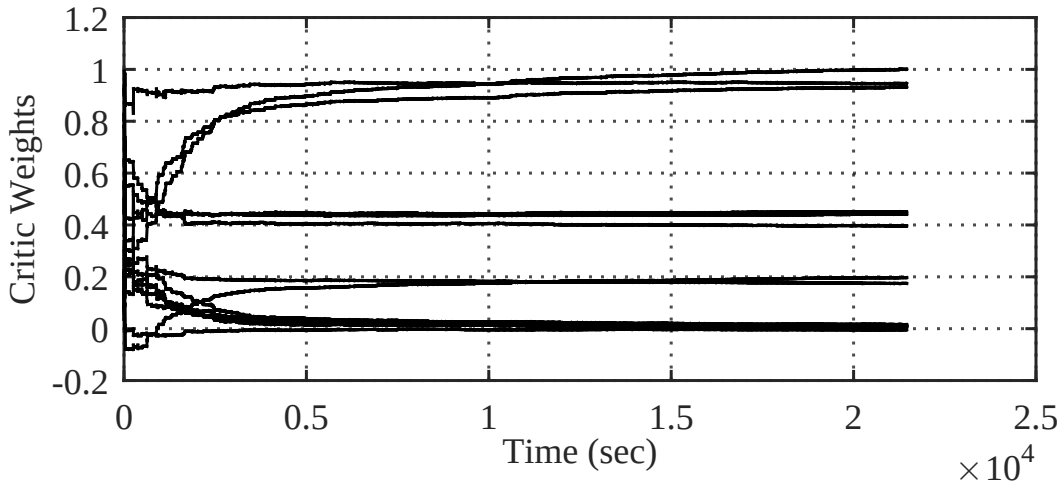

Figure 3.3: History of critic weights in online training by Algorithm 3.2.

Comparing figures 3.2 and 3.3, one can see a sharp jump in the history of weight elements in Fig. 3.2. This jump is associated to the respective weight elements which approximate the value function of the initial admissible policy. Since the selected initial policy is far from being optimal, its respective weight elements are distinctively different from the rest of the weight elements. After finding the weight elements for approximating the value function of the initial admissible policy, the history of the critic weight elements shows a smoother behavior in Fig. 3.2. On the other hand, Algorithm 3.2 does not try to solve the policy evaluation for finding the value function of selected policies and therefore, such a sharp jump does not exist in Fig. 3.3.

For evaluating the performance of concurrent training in the PI algorithm (Algorithm 3), $\alpha_0 = 300$ was selected. Both constants α_1 and α_2 were selected as 1. The concurrent training algorithm terminated in 500 seconds. The history of the critic weights in concurrent training is shown in Fig. 3.4.

For evaluating the performance of the concurrent training in the single loop PI algorithm (Algorithm 4), $\alpha_1 = 1$ and $\alpha_2 = 1$ were selected. For initializing the covariance matrix, α_0 was selected as 200. The algorithm was terminated after 50 seconds. The history of the critic weights is shown in Fig. 3.5. As one can see from Fig. 3.5 the training process shows a convergent behavior.

A notable difference between the simulation results of concurrent training in figures 3.4 and 3.5 and the online training methods shown in figures 3.2 and 3.3 is the considerable shorter time

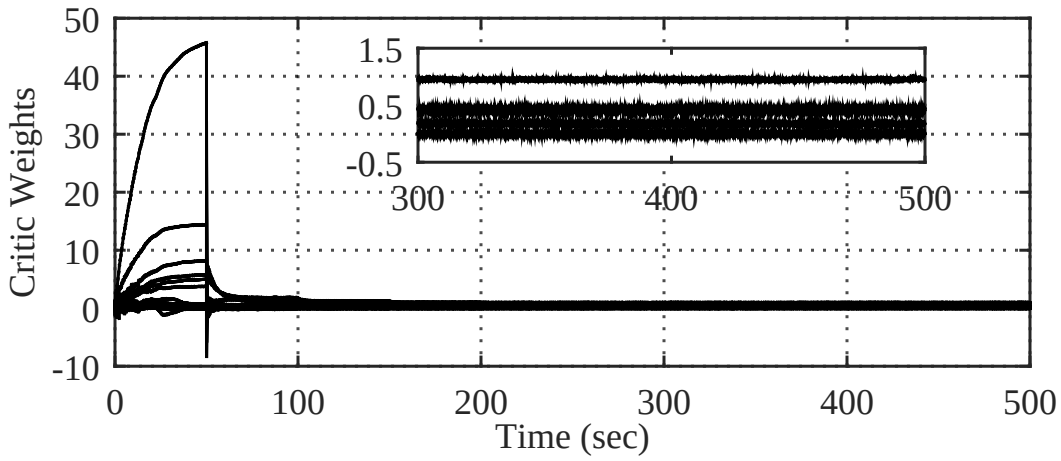

Figure 3.4: History of critic weights in concurrent training with PI algorithm (Algorithm 3)

required for training. This significant reduction in the training time shows the effect of the stored data.

A potential problem in control of switched systems is high frequency switching or even possibly infinite number of switching in a finite time. To remedy this issue, one can enforce a minimum time to be elapsed between two successive switching actions, called minimum dwell time [39]. For instance, consider system had a switching at time t_1, the dwell time is δ and system is at time instant t and needs to change its mode. In case $t - t_1 > \delta$ system can change its mode. However, if $t - t_1 \leq \delta$ system is not permitted to switch and it should continue to use the same mode that it is using until the dwell time is elapsed. Application of such a remedy without theoretically incorporating it in the design of the optimal controllers leads to sub-optimality. However, in case a small dwell time is used, the effect of the remedy will be a disturbance. Given the feedback nature of the controller, it is expected to handle small disturbances [39].

For the purpose of performance evaluation, a controller which was trained by a VI algorithm introduced in [36, 39] was simulated. In order to compare the performance of Algorithms 3.1 and 3.2 in control of the system, the controllers trained by each algorithm were used separately for regulating the system from the same initial condition. For this purpose, $x(0) = [1,1]^T$ was selected as the initial condition. The performance of the discussed controllers are compared in Fig. 3.6. As one can see from Fig. 3.6, both controllers trained by Algorithms 1 and 2 resulted in a similar

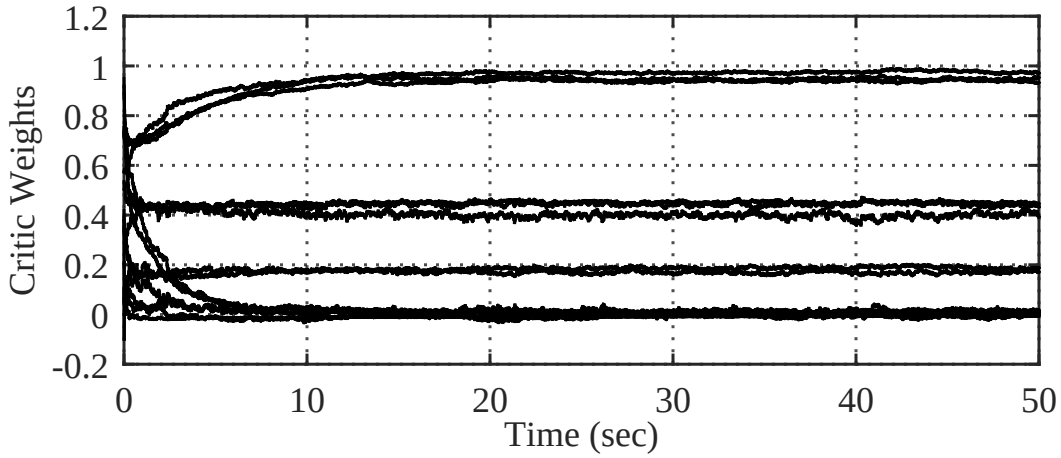

Figure 3.5: History of the critic weights in concurrent training with single loop PI algorithm (Algorithm 4).

performance in terms of state trajectories and their performance is close to that of the optimal controller reported in [36, 39].

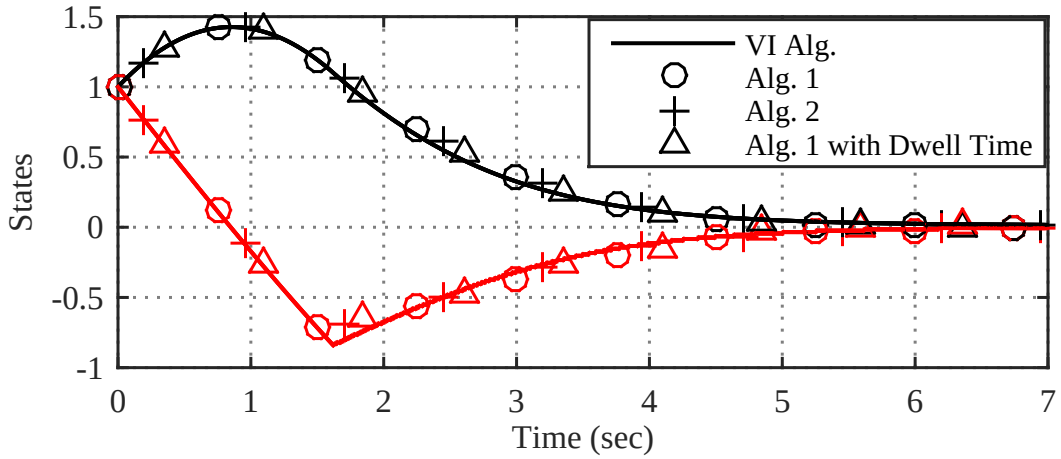

Figure 3.6: Comparison of state trajectories generated by critics trained by VI algorithm in [36], Algorithms 1 and 2 and also application of the controller trained by Algorithm 1 which operated under minimum dwell time remedy with $\delta = 0.4$ (sec). The initial condition is $x_0 = [1,1]^T$. The black signals denote the distance, i.e., $x_1(.)$ and the red signals denote the velocity, i.e., $x_2(.)$.

In order to further evaluate the performance of the concurrent training in the proposed algorithms, the previously used initial condition was used separately with the controllers trained by Algorithms 3 and 4. The state trajectories are shown in Fig. 3.7. As one can see from Fig. 3.7, both controllers resulted a similar performance and their performance is again close to that of the optimal controller

reported in [36, 39].

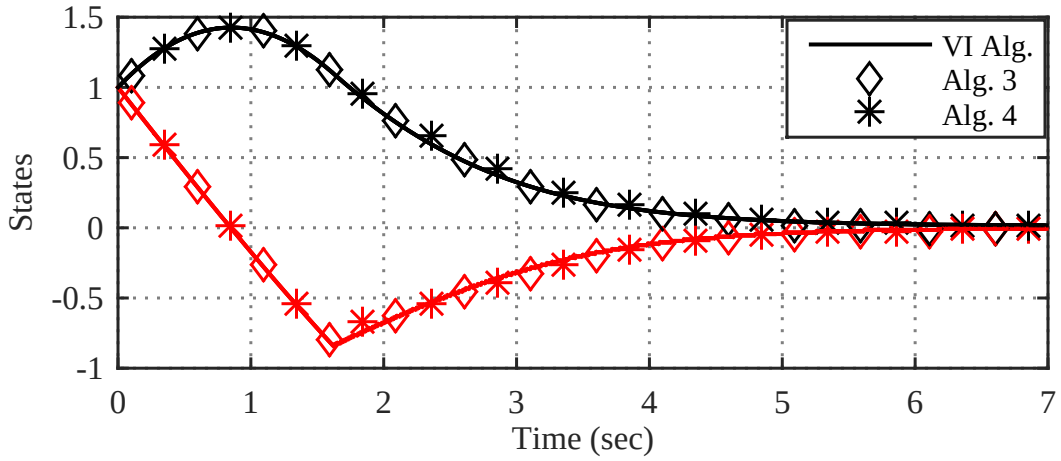

Figure 3.7: Comparison of state trajectories generated by critics trained by VI algorithm in [36], Algorithms 3 and 4. The initial condition is $x_0 = [1,1]^T$. The black signals denote the distance, i.e., $x_1(.)$ and the red signals denote the velocity, i.e., $x_2(.)$.

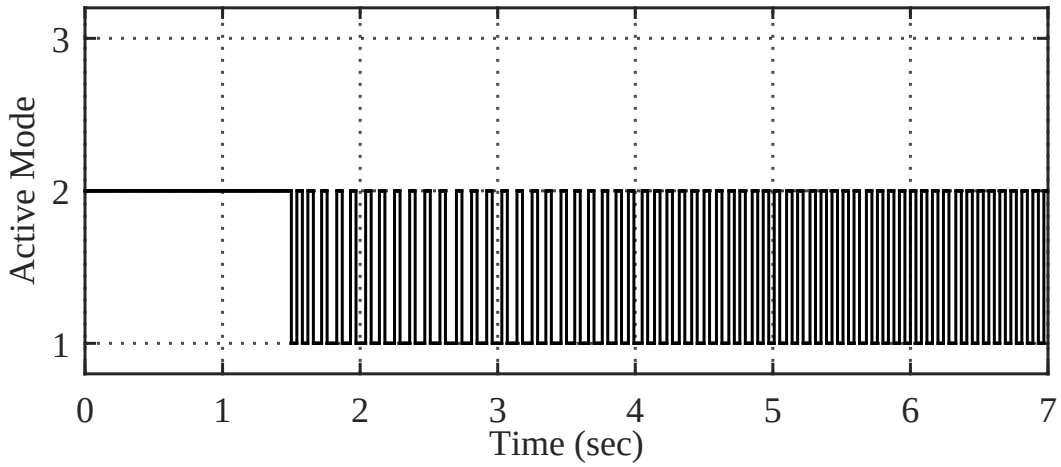

Figure 3.8: Switching schedule resulted from application of the critic trained by Algorithm 3.1 and minimum dwell time remedy with $\delta = 0.04$ (sec). The initial condition is $x_0 = [1,1]^T$.

To show the effect of minimum dwell time remedy, the controller which was trained by Algorithm 1 was chosen to regulate the states from the same initial condition that was used in the previous examples. $\delta = 0.04$ (sec) was selected as the dwell time. The history of the states and the switching schedule are shown in Fig. 3.6 and 3.8. For a better comparison, the switching schedule without minimum dwell time remedy is also shown in Fig. 3.9. As one can see from Fig. 3.6, the

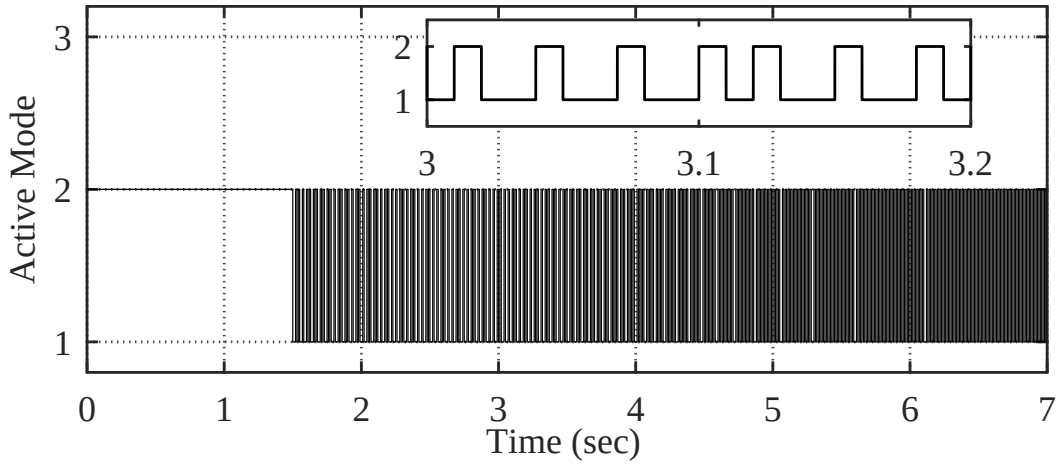

Figure 3.9: Switching schedule resulted from application of the critic trained by Algorithm 3.1 without minimum dwell time remedy. The initial condition is $x_0 = [1, 1]^T$.

system had a good performance in terms of state regulation while no high frequency switching can be seen in Fig. 3.8. However, high frequency switching can be easily seen in Fig. 3.9.

3.7. Conclusion

Sub-optimal scheduling in switched systems with autonomous subsystems and continuous-time dynamics was studied. Two solutions were established based on an approximate dynamic programming approach. The first solution formulated a classic policy iteration algorithm with recursive least squares training law. In order to relax the computational effort in the algorithm, another solution was established which aimed at finding the optimal value function directly. Since the second algorithm did not include solving a Lyapunov equation for generating the evolving policies, resulting immature policies were not necessarily stabilizing which necessitated the presence of a supervisory stabilizing policy for online training. Online and concurrent training algorithms were discussed for implementing each solution. Numerical simulations were conducted to verify the effectiveness of the presented algorithms. An important concern in designing optimal schedulers for switched systems is how the controllers handle high frequency switching. The effect of a remedy to prevent high frequency switching was shown through numerical simulations in this chapter. However, further theoretical investigations for designing similar optimal controllers with such a

capability is subject of the future work.

Chapter 4

Near-optimal Switching in Anti-lock Brake Systems using Approximate Dynamic Programming

Optimal scheduling in anti-lock brake system of ground vehicles is performed through approximate dynamic programming for reducing the stopping distance in severe braking. The proposed optimal scheduler explicitly incorporates the hybrid nature of anti-lock brake system and provides a feedback solution with a negligible computational burden in control calculation. To this goal, an iterative scheme, called value iteration algorithm, is used to learn the infinite horizon solution to the underlying Hamilton-Jacobi-Bellman equation. Performance of the proposed method in control of the brake system is illustrated using both linear-in-parameter neural networks and multi-layer Perceptrons. Simulation results demonstrate potentials of the method.

Index Terms- optimal switching, approximate dynamic programming, anti-lock brake system.

4.1. Introduction

An Anti-lock Brake System (ABS) is an important active safety system which prevents wheel lock-ups during severe braking [32, 57]. By preventing wheel lock-ups, the friction coefficient between the wheel and road surface increases which results in generation of greater braking torque compared to that of an ordinary brake system, leading to a shorter stopping distance [25, 70]. Meanwhile, by preventing wheel lock-ups, ABS can increase directional stability and steering ability of the vehicle during severe braking [72, 73].

In control of ABS, the objective is operating the system such that wheel lock-up is prevented. This task can be performed in two major methods. The first method uses thresholding control logic based on data gathered from wheel angular velocity and deceleration measurements [23, 106]. Hence, when the wheel velocity is within the lock-up threshold, the controller reduces the braking pressure and releases the wheel [23]. This method requires a time consuming calibration process for the deceleration thresholds [25] and suffers from being inefficient since it neglects the vertical

load and magnitude of friction force in its thresholding logic [23, 99, 106]. A more efficient way to prevent wheel lock-ups in braking is provided through regulating the longitudinal wheel *slip* ratio. Slip ratio, denoted typically in the literature by λ, is the ratio of the difference between wheel linear velocity and vehicle velocity over vehicle velocity. By definition, $0 \leq \lambda \leq 1$ where the lower and upper limits show no slip and complete sliding, respectively [17, 21, 70, 82].

The relationship between the friction coefficient and slip ratio is highly nonlinear and depends on the road conditions [57, 70]. However, for most of road conditions the maximum of friction coefficient achieves at $\lambda \in [0.15, 0.2]$ which gives the desired slip ratio. While online identification of the road condition is still an open research problem [25], the control policy in ABS tries to regulate the slip ratio around the desired value. For this goal, automotive industry is widely using rule-based controllers along with elaborating look-up tables [84]. Lack of robustness, time consuming calibration process, high level of complexity in analyzing these tables [84] and increasing demand for more reliable braking systems in modern vehicles have led to huge number of research studies for new ABS controllers [13, 16, 17, 21, 22, 25, 41, 46, 60, 63, 65, 72, 73, 85, 99, 107, 108, 115].

Some novel brake systems have been studied recently for implementing ABS [115, 127, 133]. In conventional vehicles ABS is typically implemented through hydraulic brake systems with solenoid valves which can only be on or off [57]. The on/off configuration of the solenoid valves generates three *modes* which corresponds to increasing, decreasing and holding the braking torque [70]. In general, this combination of continuous time subsystems with discrete time events (e.g. switching in ABS) portrays a hybrid dynamical system [126, 136]. More specifically, the system is a switched system. Control of switched systems by scheduling among their modes is a challenging task due to discrete nature of the control signal.

As a powerful control method, optimal control has always attracted the attentions. The objective of optimal control is designing a control signal which minimizes a cost function subject to state/control constraints [58]. In general, the problem of optimal control reduces to solving the so-called Hamilton-Jacobi-Bellman (HJB) equation which provides the necessary and sufficient condition for optimality. A systematic and closed form solution for optimal control is provided by Dynamic Programming (DP). However, DP only provides backward-in-time offline solution and it

suffers from the so-called curse of dimensionality and modeling in high order systems [58, 62]. In order to derive an online solution which avoids the mentioned curses in DP, Approximate Dynamic Programming (ADP) is suggested [8, 116, 119, 121]. In summary, ADP uses function approximators, mostly neural networks, to approximate the cost-to-go (value function). The key idea is tuning the parameters of the mentioned function approximators through iterative schemes to find the optimal value function which solves the HJB equation. In ADP literature, this process is generally called *training*. One of the iterative schemes widely used in ADP is Value Iteration (VI) algorithm [124]. An important feature of VI algorithm is its simple recursion which leads to negligible computational burden. For the case of switched systems, VI algorithm has been used for control of switched systems with autonomous subsystems and finite horizon cost function [39], infinite horizon cost function [36], and switched systems with homogenous subsystems [91, 92].

In ADP literature, value function approximation is mostly performed by Linear-in-Parameter (LIP) neural networks. Application of LIP neural networks leads to simplicity in training with least squares methods and low computational burden in online evaluation of the trained networks. Theoretically, one can improve the approximation precision of LIP neural networks to any desired degree by increasing the number of basis functions, *neurons*, to infinity [93]. Unfortunately in practice one cannot increase neurons after a certain number in offline batch mode training with least squares due to computational concerns. This problem leads to poor value function approximation and poor performance of the closed loop system accordingly. Hence, it is desired to improve the approximation precision in value function approximation for improving the performance of the control system.

From switching control point of view, one can categorize the presented control methods for ABS in two major groups. In the first group, the switching nature of ABS is mostly neglected. Neglecting the switching nature of ABS, its control reduces to application of conventional nonlinear control methods [72, 73, 89, 99, 135]. A more challenging approach is incorporating the switching nature of ABS as reported in [13, 16, 18, 22, 33, 41, 44, 46, 70, 81]. Although the mentioned studies consider the switching nature of ABS, their solutions are not optimal. The optimality in ABS was studied in [3, 47, 61, 71, 74, 106]. Except the gain scheduling in [47] and threshold method in [71],

to the best of the knowledge of the authors of the present study, the switching nature of ABS is mostly neglected in seeking the optimal solutions. Meanwhile, the proposed optimal solutions which deal with switching dynamics do not solve explicitly for optimal switching, for instance through thresholding with fuzzy systems or linearization and gain scheduling, which adds another degree of approximation error in defining the active mode. In this chapter, value iteration algorithm for finding the optimal switching schedule in ABS is studied. Three important contributions of the present study are as follows[1].

- The present study directly deals with the switching nature of ABS.
- The proposed method in this chapter solves for the optimal switching solution.
- The performance of VI algorithm in optimal switching problems is improved.

For improving the approximation precision of the value function approximator, Multi-Layer Perceptron (MLP) neural networks with sufficiently rich hidden layers are suggested. For offline training of the MLP networks in batch mode, Bayesian regularization back-propagation algorithm is used and the results of using LIP and MLP neural networks for value function approximation are compared. Meanwhile, to ensure the suitable performance of the proposed controller in practical implementations, application of minimum dwell time remedy is studied to prevent high frequency switching. According to simulation results, application of MLP neural networks with sufficiently rich hidden layers leads to significant improvements in the performance of the optimal controller in terms of regulating the slip ratio and decreasing the stopping distance.

The rest of this chapter is organized as follows. In section 4.2, the hydraulic brake system model is discussed. The controller design is discussed in section 4.3. Value function approximation and implementing VI algorithm to find the optimal switching schedule are discussed in section 4.4. Simulation results are presented in section 4.5. At last, section 4.6 concludes the chapter.

4.2. Modeling

[1]The preliminary results of this chapter was presented in ASME 2015 Dynamic Systems and Control Conference (*DSCC*2015) [94]. Due to large approximation errors in approximating the value function, the controller showed a poor performance on-the-fly in [94]. This problem is addressed in the present study.

Table 4.1: Nomenclature

T_b	Braking Torque $(N.m)$	C_1, C_2 & C_3	Coefficients of Burckhardt Model
M	Quarter of Vehicle Mass $(\mathbf{k}g)$	μ_H	Constant in Friction Model
g	Acceleration due to Gravity (m/s^2)	A_{wc}	Wheel Cylinder Area (m^2)
ω	Wheel Angular Velocity (rad/s)	η_{wc}	Mechanical Efficiency
F_f	Longitudinal Force (N)	B_f	Braking Factor
F_Y	Surface Reaction Force (N)	r_r	Effective Radius of Braking Disk (m)
$\mathcal{X}$	Vehicle Traveled Distance(m)	K_b	Braking Torque Coefficient (m^3)
$\mathcal{V}$	Vehicle Velocity (m/s)	M_w	Mass of the Wheel $(\mathbf{k}g)$
R	Wheel Radius (m)	h_{cg}	Height of Vehicle's Center of Mass (m)
I	Wheel Moment of Inertia $(\mathbf{k}g.m^2)$	l_{wb}	Length of Wheel Base (m)
λ	Slip Ratio	$V(.)$	Cost-to-go
P	Braking Pressure $(\mathbf{k}Pa)$	$V^*(.)$	Optimal Cost
A_1 & A_2	Constants of Orifice Area (m^2)	γ	Discount Factor
C_{d_1} & C_{d_2}	Coefficients of Built/Dump Valves	t	Time (s)
C_f	Coefficient of Brake Fluid Flow $(m^{\frac{9}{2}}.s/\mathbf{k}g)$	k	Discrete Time Index
P_p	Pump Pressure $\mathbf{k}Pa$	δt	Discretization Sample Time (s)
P_r	Reservoir Pressure $(\mathbf{k}Pa)$	Δt	Dwell Time (s)
ρ	Brake Fluid Density $(\mathbf{k}g/m^3)$	i	Learning Algorithm Iteration Index
μ	Friction Coefficient		

For Modeling the brake system dynamics, quarter vehicle model (single corner), half vehicle model, and 4 wheel vehicle model [90] are available. In this study, a single corner vehicle model is selected [57, 70]. To avoid unnecessary complexities and nonlinearities in the model, the steering effects and drag forces are neglected. Meanwhile, it is assumed that each wheel carries quarter of the total load. The brake system dynamics can be presented through 4 major equations corresponding to vehicle traveled distance, vehicle velocity, slip ratio and braking pressure. These equations can be directly derived from the physics of the brake system. The free body diagram of a single wheel is shown in Fig. 4.1. In Fig. 4.1, T_b denotes the braking torque $(N.m)$, Mg is quarter of

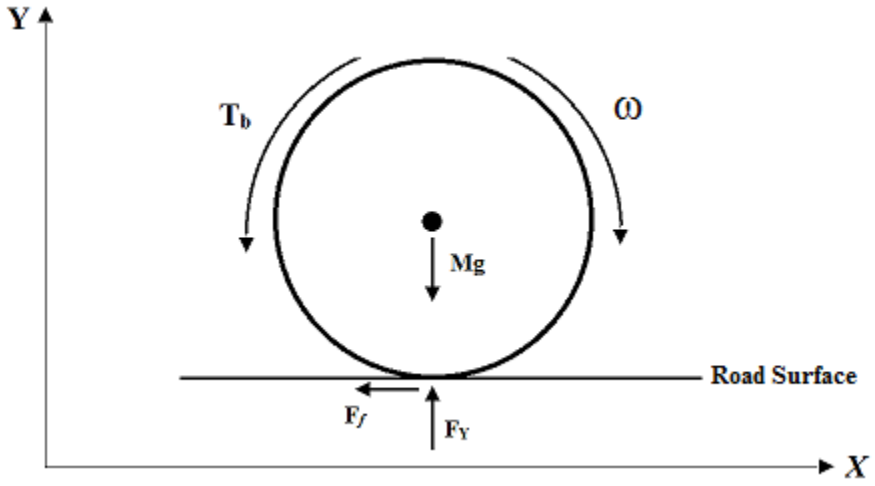

Figure 4.1: Free body diagram of a single wheel [57].

the total vehicle weight $(\mathbf{k}g.m/s^2)$, ω is the angular velocity of the wheel (rad/s), and F_f is the friction/longitudinal force acting on the wheel (N). F_Y is the surface reaction force from the road surface (N). Considering the forces along the X-axis with positive direction denoted in Fig. 4.1, by applying Newton's second law one has

$$\sum F_x = M\ddot{\mathcal{X}}(t) \Rightarrow -F_f(\lambda(t)) = M\dot{\mathcal{V}}(t) \tag{4.1}$$

In Eq. (4.1), M denotes quarter of the vehicle total mass $(\mathbf{k}g)$, $\mathcal{X}(.)$ denotes the vehicle traveled distance (m), and $\mathcal{V}(.)$ is the vehicle velocity (m/s). $\lambda(.)$ is the slip ratio and t denotes the time (s).

Meanwhile, by selecting the clockwise direction as the positive direction and applying Newton's second law for the moments, one has

$$\sum T = I\dot{\omega}(t) \Rightarrow -T_b(t) + R \times F_f(\lambda(t)) = I\dot{\omega}(t)? \tag{4.2}$$

where R is the wheel radius (m) and I denotes the wheel moment of inertia $(\mathbf{k}g.m^2)$. Since the purpose of the controller is controlling the wheel slip ratio, it is also necessary to model the slip ratio as

$$\lambda(t) = \frac{\mathcal{V}(t) - R\omega(t)}{\mathcal{V}(t)} \tag{4.3}$$

By differentiating both sides of Eq. (4.3) with respect to time and substituting from equations (4.1) and (4.2), one has

$$\dot{\lambda}(t) = \frac{-1}{\mathcal{V}(t)}[\frac{F_f(\lambda(t))}{M}(1-\lambda(t)) + \frac{R^2 F_f(\lambda(t))}{I}] + \frac{RT_b(t)}{\mathcal{V}(t)I} \tag{4.4}$$

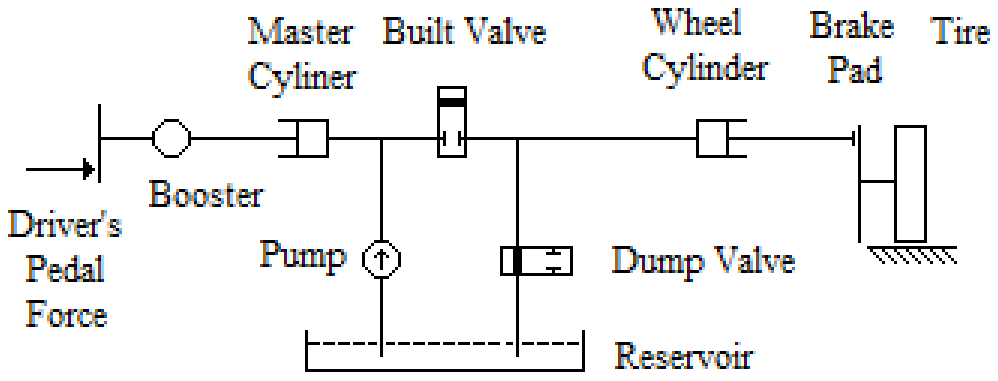

Figure 4.2: Structure of a standard hydraulic brake system showing the pressure decrease as the brake fluid is flowing to the reservoir from the brake line through the open dump valve [23].

The next equation describes the dynamics of the braking pressure. The structure of a standard hydraulic brake system is shown in Fig. 4.2. In Fig. 4.2, built and dump valves are solenoid valves which can only be on or off. Based on the on/off configuration of built and dump valves, the braking pressure can increase, decrease or hold constant as follows. If built value is open and dump valve is closed, the pressure increases. On the other hand, if built valve is closed and dump valve is open, the pressure decreases. In case both valves are closed, the pressure would be held constant. The condition in which both valves are open is not permitted [23, 46, 84]. For modeling the braking pressure, the flow of brake fluid in the hydraulic circuit is considered as flow of fluid in an orifice as [23]

$$\dot{P}(t) = \frac{A_1 C_{d_1}}{C_f}\sqrt{\frac{2}{\rho}(P_p - P(t))} - \frac{A_2 C_{d_2}}{C_f}\sqrt{\frac{2}{\rho}(P(t) - P_r)} \tag{4.5}$$

In Eq. (4.5), $P(.)$ denotes the braking pressure ($\mathbf{k}Pa$). A_1 and A_2 are constants representing the orifice area. C_{d_1} and C_{d_2} are coefficients of built and dump valves which can be 0 or 1 and C_f is the coefficient of brake fluid flow ($m^{\frac{9}{2}}.s/\mathbf{k}g$). P_p is the constant pump pressure ($\mathbf{k}Pa$), and P_r is the constant reservoir pressure ($\mathbf{k}Pa$). At last, ρ is the density of brake fluid in the hydraulic brake system ($\mathbf{k}g/m^3$).

Equations (4.1), (4.2), (4.4), and (4.5) form the major dynamics which are used in this study to model ABS for control. The relations for friction coefficient, longitudinal force and braking torque are represented as follows. For friction coefficient, Burckhardt friction model is selected where $\mu(.)$ can be represented as [84, 106, 115]

$$\mu(\lambda(t)) = \mu_H(C_1(1 - e^{-C_2\lambda(t)}) - C_3\lambda(t)) \tag{4.6}$$

where μ_H is a constant. The coefficients C_1, C_2, and C_3 depend on the road conditions. Interested readers are referred to [84] for a comprehensive discussion on these constants for different road conditions. The longitudinal force, $F_f(.)$, can be modeled as

$$F_f(\lambda(t)) = \mu(\lambda(t))F_Y \tag{4.7}$$

where F_Y is the surface reaction force (N) as defined earlier.

At last, the braking torque in Eq. (4.2) can be given by [23]

$$T_b(t) = (A_{wc}\eta_m B_f r_r)P(t) = K_b P(t) \tag{4.8}$$

In Eq. (4.8), A_{wc} is the wheel cylinder area (m^2), η_m is the mechanical efficiency, B_f is the braking factor, r_r is the effective radius of the braking disk (m), and finally K_b is the braking torque coefficient (m^3).

For state-space representation of the brake system, state variables are selected as vehicle traveled distance, vehicle velocity, slip ratio, and braking pressure, i.e. $x(t) = [x_1(t), x_2(t), x_3(t), x_4(t)]^T = [\mathcal{X}(t), \mathcal{V}(t), \lambda(t), P(t)]^T$ where superscript T denotes the transpose operator. Hence, one has

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{-F_f(x_3(t))}{M} \\
\dot{x}_3(t) &= \frac{-1}{x_2(t)}\left[\frac{F_f(x_3(t))}{M}(1 - x_3(t)) + \frac{R^2 F_f(x_3(t))}{I}\right] + \frac{RK_b}{x_2(t)I}x_4(t) \\
\dot{x}_4(t) &= m_{vc}(x(t)), vc \in \{1,2,3\}
\end{aligned} \tag{4.9}$$

The first three state equations in Eq. (4.9) do not depend on the configuration of the solenoid valves. However, the last state equation includes the braking pressure which leads to the three cases of increasing, decreasing and holding the braking pressure. Hence, the active mode can be one of the following modes.

$$\begin{cases}
m_1(x(t)) = \frac{A_1}{C_f}\sqrt{\frac{2}{\rho}(P_p - x_4(t))} & \text{Increases: } C_{d_1} = 1, C_{d_2} = 0 \\
m_2(x(t)) = -\frac{A_2}{C_f}\sqrt{\frac{2}{\rho}(x_4(t) - P_r)} & \text{Decrease: } C_{d_1} = 0, C_{d_2} = 1 \\
m_3(x(t)) = 0 & \text{Hold: } C_{d_1} = 0, C_{d_2} = 0
\end{cases} \tag{4.10}$$

At any time instant t, only one mode is active in Eq. (4.10). For notational simplicity, one can refer to the dynamics of ABS as

$$\dot{x}(t) = f_{vc}(x(t)), vc \in \{1,2,3\} \tag{4.11}$$

Remark 4.2.1 The existence of vehicle velocity in the denominator of $\dot{x}_3(.)$ in Eq. (4.9) leads to large magnitude of $\dot{\lambda}(.)$ in small quantities of vehicle velocity and divergence of Eq. (4.9), accordingly. In order to prevent divergence in the brake system model, one can switch off the ABS and use the ordinary brake system in sufficiently safe low velocities, for instance $V \leq 5$ (m/s) [46, 47, 70]. Another approach is based on the physics of the problem. Since $\lambda > 1$ is not physically justifiable in braking, one can add a saturation unit in the dynamics such that at any time wheel locks-up, the saturation unit will set λ to 1. □

Remark 4.2.2 The effect of lateral movement in braking is neglected in this chapter and only braking in straight lines is studied. Hence, the longitudinal friction model that is introduced in Eq. (4.7) does not consider the lateral slip. For combined slip, one can use Dugoff friction model which addresses both longitudinal and lateral slip ratios [24, 105]. □

Remark 4.2.3 For deriving the surface reaction force, F_Y, one needs to consider the vertical forces. The vertical forces are comprised of a static force due to vehicle total weight and a dynamic force due to load transfer during braking because of suspension. Hence, one has [73]

$$F_Y(t) = Mg - \frac{(M - M_w)h_{cg}}{2l_{wb}}\ddot{\mathcal{X}}(t) \tag{4.12}$$

In Eq. (4.12), M_w is mass of the vehicle wheel $(\mathbf{k}g)$, h_{cg} is height of vehicle's center of mass (m), and l_{wb} is length of wheel base (m). Assuming the positive vehicle velocity in the direction of vehicle motion, in braking, vehicle velocity is decreasing which leads to $\ddot{\mathcal{X}}(.)$ being negative. Hence, the load transfer component of $F_Y(.)$ makes the total vertical force on the front wheel slightly greater

than the quarter of the vehicle weight. Considering the plot of vehicle velocity in braking, one can see that velocity decreases with a somehow constant slope. Meanwhile, existence of $\ddot{X}(.)$ in the dynamic component adds to the modeling complexities of the system which is not desired. Hence, in this study, the effect of load transfer is not considered in modeling $F_Y(.)$ and this effect is considered as a modeling uncertainty. □

4.3. Controller Design

The purpose of this section is introducing a closed form solution for finding the optimal switching scheduler in ABS. This scheduler assigns the switching instants and appropriate active modes such that a performance index is minimized. In fact, the proposed controller tries to learn the infinite horizon solution of the underlying HJB equation through an iterative scheme called VI algorithm. VI algorithm can be formulated for both continuous-time and discrete-time dynamics. In this study, the discrete-time format of this algorithm is studied. Hence, for designing the controller, the first step is discretizing the dynamics in the time domain. For this goal, Euler integration method is used. Denoting discretization sample time by δt and discrete time index by k one has

$$x_{k+1} = x_k + \delta t \dot{x}_k \tag{4.13}$$

where $x_k \in \mathbb{R}^n$ is the state vector x at time k, i.e. $x_k \equiv x(k)$. Considering the dynamics of ABS as represented in Eq. (4.11), substituting $\dot{x}_k$ by $f_{vc}(x_k)$ in Eq. (4.13), one has $x_{k+1} = x_k + \delta t f_{vc}(x_k)$. Letting $f_{v(x_k)}(x_k) \equiv x_k + \delta t f_{vc}(x_k)$

$$x_{k+1} = f_{v(x_k)}(x_k) \tag{4.14}$$

For developing the optimal solution, one needs to define the cost function. The infinite horizon cost function can be defined as

$$J(x_0) = \sum_{\bar{k}=0}^{\infty} \gamma^{\bar{k}} (x_{\bar{k}} - x_d)^T Q (x_{\bar{k}} - x_d) \tag{4.15}$$

where $0 < \gamma \leq 1$ is the discount factor and $x_d \in \mathbb{R}^n$ shows the desired states. $Q \in \mathbb{R}^{n \times n}$ is called state penalizing matrix which is a design parameter chosen as a positive semi-definite matrix. Based

on the definition of the cost function in Eq. (4.15), one can define the cost-to-go, $V:\mathbb{R}^n \to \mathbb{R}$, as the cost of going from time t, or equivalently discrete time index k, to ∞ as

$$\begin{aligned} V(x_k) &= \sum_{\bar{k}=k}^{\infty} \gamma^{\bar{k}-k}(x_{\bar{k}}-x_d)^T Q(x_{\bar{k}}-x_d) \\ &= (x_k-x_d)^T Q(x_k-x_d) + \gamma \sum_{\bar{k}=k+1}^{\infty} \gamma^{\bar{k}-(k+1)}(x_{\bar{k}}-x_d)^T Q(x_{\bar{k}}-x_d) \\ &= (x_k-x_d)^T Q(x_k-x_d) + \gamma V(x_{k+1}) \end{aligned} \tag{4.16}$$

Considering x_{k+1} as Eq. (4.14), one can further simplify Eq. (4.16) as

$$V(x_k) = (x_k-x_d)^T Q(x_k-x_d) + \gamma V(f_{v(x_k)}(x_k)) \tag{4.17}$$

Equation (4.16) simply means that the cost of going from time k to ∞ is equal to the sum of costs of going from time k to time $k+1$ and cost from time $k+1$ to ∞. Based on Bellman principle of optimality, one can define the optimal cost-to-go, $V^*:\mathbb{R}^n \to \mathbb{R}$, from Eq. (4.17) as

$$\begin{aligned} V^*(x_k) &= (x_k-x_d)^T Q(x_k-x_d) + \gamma V^*(x^*_{k+1}) \\ &= (x_k-x_d)^T Q(x_k-x_d) + \gamma V^*(f_{v^*(x_k)}(x_k)) \end{aligned} \tag{4.18}$$

where

$$v^*(x_k) = \arg\min_{w\in\mathbb{V}} \Big(V^*(f_w(x_k)) \Big) \tag{4.19}$$

In Eq. (4.19), $\mathbb{V}$ is the set of all possible modes in the system. Equation (4.18) is the discrete-time HJB equation for switched systems with autonomous subsystems. If $V^*(.)$ is known, one can easily use the feedback scheduler introduced in Eq. (4.19) for online control with a small computational burden in feedback calculations. This computational burden comprised of calculating the right-hand side of Eq. (4.19) for all existing modes and choosing the mode which leads to the minimum [36,39]. However, solving the HJB equation for $V^*(.)$ is very difficult and in many cases impossible [8,62]. One convenient solution for this problem is using iterative schemes which will converge to the solution of the HJB equation. This idea is adapted in this study. For this purpose, VI algorithm

from approximate dynamic programming techniques is used. VI algorithm has two stages which are called policy evaluation and policy update. In the policy evaluation, immature value function is updated along existing policy as

$$V^{i+1}(x_k) = (x_k - x_d)^T Q(x_k - x_d) + \gamma V^i(f_{v^{i+1}(x_k)}(x_k)) \tag{4.20}$$

where superscript i is the iteration index of VI algorithm. From Eq. (4.20), one can see that $V^{i+1}(.)$ can be found from $V^i(.)$ in VI algorithm. This characteristic of VI algorithm is known as partial back-up which enables VI algorithm to have a simple recursion and negligible computational burden accordingly [8]. Meanwhile, policy update stage can be represented as

$$v^{i+1}(x_k) = \underset{w \in \mathbb{V}}{\arg\min} \left(V^i(f_w(x_k)) \right) \tag{4.21}$$

The algorithm starts by choosing an initial value function, $V^0(.)$. With $V^0(.)$ and policy update as in Eq. (4.21), one finds $v^1(.)$. Afterward, with $V^0(.)$, $v^1(.)$ and Eq. (4.20), one will find $V^1(.)$. Again, with $V^1(.)$ and Eq. (4.21), one can find $v^2(.)$. With a similar procedure, this iterative process continues until no meaningful difference can be detected between two successive value functions generated by Eq. (4.20).

Remark 4.3.1 By a close look at the definition of the cost function in Eq. (4.15) and the cost-to-go in Eq. (4.16), one can see that in both equations, vehicle traveled distance, as one of the states, exists in the infinite horizon series. Noting that vehicle traveled distance will not lead to zero as the vehicle stops, one can see that both infinite horizon cost function and cost-to-go do not satisfy the necessary condition for convergence of series [93] and will blow-up. In order to solve this problem, a discount factor, γ is used in the definition of the cost function. By using a discount factor the cost will converge and one can approximate this cost accordingly. Considering time instant k, existence of small discount factor shows that more emphasize is devoted to the recent data to calculate the cost rather than the whole infinite horizon [8, 62]. □

4.4. Value Function Approximation and Implementation of VI Algorithm

One of the challenges in implementing VI algorithm is finding $V^{i+1}(x_k)$ such that $\forall x_k \in \Omega$, Eq. (4.20) is satisfied where $\Omega \subset \mathbb{R}^n$ is a compact region of training which includes the origin. Assuming value function as a differentiable function, one can use neural networks to uniformly approximate $V^{i+1}(.)$ [42]. The common choice of neural networks in the ADP literature for value function approximation are LIP neural networks with their basis vector comprised of linearly independent polynomials. Theoretically, this kind of neural network can approximate any continuously differentiable function to any degree of precision based on Weierstrass approximation theorem [93]. Considering the optimal value function $V^*(.)$, one can define the exact reconstruction as

$$V^*(x_k) = \sum_{j=1}^{\infty} W_{LIP\,j}\phi_j(x_k) \tag{4.22}$$

where $W_{LIP\,j} \in \mathbb{R}$ are the coefficients and $\phi_j : \mathbb{R}^n \to \mathbb{R}$ are the polynomial basis functions. However, in practice the infinite summation is not considered. Mostly, a finite number of basis functions are selected and the value function is considered as

$$V^*(x_k) = W_{LIP}^{*T}\phi(x_k) + \varepsilon(x_k) \tag{4.23}$$

In Eq. (4.23), $W_{LIP}^* \in \mathbb{R}^{m_{LIP}}$ represents the weight vector where m_{LIP} is the number of basis functions (*neurons*). $\phi : \mathbb{R}^n \to \mathbb{R}^{m_{LIP}}$ is a vector of linearly independent basis functions and $\varepsilon : \mathbb{R}^n \to \mathbb{R}$ is the approximation error which represents the tail of the infinite summation after truncating it. Based on Eq. (4.23), one has

$$\widehat{V}_{LIP}(x_k) = \widehat{W}_{LIP}^T\phi(x_k) \tag{4.24}$$

where $\widehat{V} : \mathbb{R}^n \to \mathbb{R}$ is the approximate value function and $\widehat{W}_{LIP} \in \mathbb{R}^{m_{LIP}}$ is the tunable weight vector. In general, one is interested in finding $\widehat{W}_{LIP}$ such that $\widehat{V}_{LIP}(.)$ in Eq. (4.24) approximates $V^*(.)$ in Eq. (4.23). As mentioned before, this process is called training. Based on the presented value function approximation in Eq. (4.24), one can define the policy evaluation and policy update in VI algorithm

as

$$\widehat{W}_{LIP}^{i+1^T}\phi(x_k) = (x_k - x_d)^T Q(x_k - x_d) + \gamma \widehat{W}_{LIP}^{i^T}\phi(f_{v^{i+1}}(x_k)) \tag{4.25}$$

$$v^{i+1}(x_k) = \arg\min_{w \in \mathbb{V}} \left(\widehat{W}_{LIP}^{i^T}\phi(f_w(x_k)) \right) \tag{4.26}$$

Linear-in-parameter neural networks have a simple structure which results in significantly small computational burden in online applications. Meanwhile, by choosing linearly independent basis functions, one can easily apply least squares methods for training these networks. In practice, however, application of LIP neural networks for approximating the value functions demonstrates poor performance especially when the domain of training increases. Hence, application of neural networks with more precise performance compared to that of LIP networks for approximating the value function is desirable.

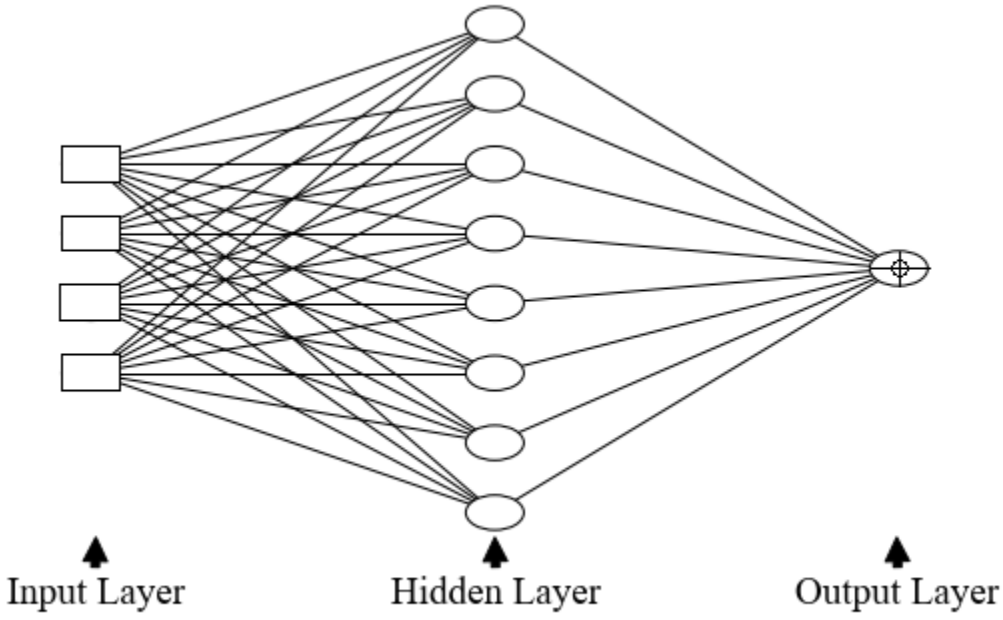

Figure 4.3: Structure of a MLP neural network with 1 hidden layer [34].

To improve the approximation precision, one can use neural networks with universal approximation capability such as MLP neural networks [34]. The structure of a MLP neural network with

one hidden layer is illustrated in Fig. 4.3. As shown in Fig. 4.3, MLP neural networks have three types of layers namely input layer, hidden layer, and output layer. In MLP neural networks, each two successive layers are connected to each other through some weight and bias terms which need to be adjusted through the training process. The input layer is connected to the environment outside the network and receives the input of the network. The hidden layer is equipped with nonlinear activation functions which perform the nonlinear mapping. The output of the network is generated through the output layer which has its own activation functions [34]. In MLP neural networks, the number of activation functions in each layer defines the number of neurons in that respective layer, m_{MLP}.

Consider the MLP network shown in Fig. 4.3 with m_{MLP} number of neurons in the hidden layer and one neuron in the output layer. Considering the input vector of the neural network as $x_k \in \mathbb{R}^n$, the working mechanism of this network is as follows. The input vector is first multiplied by the weight matrix $W^1_{MLP} \in \mathbb{R}^{m_{MLP} \times n}$ and the result is added to the bias vector of the first hidden layer $b^1_{MLP} \in \mathbb{R}^{m_{MLP}}$ making $net_1(x_k) = W^1_{MLP} x_k + b^1_{MLP}$. Assuming the activation functions in the hidden layer as $\mathcal{F}(.)$, this function acts on $net_1(x_k)$ and the result is multiplied by the weight vector of the output layer $W^2_{MLP} \in \mathbb{R}^{1 \times m_{MLP}}$ and added to output layer bias term $b^2_{MLP} \in \mathbb{R}$. Assuming the activation function in the output layer as $\mathcal{G}(.)$ the output of the MLP network for value function approximation can be defined as

$$\widehat{V}_{MLP}(x_k) = \mathcal{G}\Big(W^2_{MLP}(\mathcal{F}(W^1_{MLP} x_k + b^1_{MLP})) + b^2_{MLP}\Big) \tag{4.27}$$

Based on policy evaluation and policy update equations presented in equations (4.20) and (4.21), and also the choice of neural networks for approximating the value function presented by equations (4.24) and (4.27), one can present the VI algorithm as Algorithm 4.1.

Algorithm 4.1 *VI Algorithm (Offline Training, Batch Mode)*

step 1: *Set $i = 0$ and generate η random training samples $x^{[l]} \in \Omega$ where $l \in \{1, 2, ..., \eta\}$ and η is sufficiently large number.*

step 2: *Pick an initial value function, $\widehat{V}^0(.)$, through initial choice of your neural network parame-*

ters and find policy, $v^1(.)$, from Eq. (4.21).

step 3: *Substitute all the training samples in the right-hand side of Eq. (4.20).*

step 4: *Train the neural network and find the parameters to approximate $\widehat{V}^{i+1}(.)$ form Eq. (4.20). Set $i = i+1$.*

step 5: *Update the policy using Eq. (4.21).*

step 6: *Go back to step 3 until the absolute difference between two successive parameters of the trained neural networks is less than a selected small positive convergence tolerance, β.*

step 7: *Set $V^*(.) = \widehat{V}^i(.)$ and stop training.*

Remark 4.4.1 As mentioned before, for offline training of LIP neural network in batch mode one can easily use the least squares method in step 4 of *Algorithm* 4.1. The details of offline training with least squares was presented in appendix of [39]. For training the MLP networks, one can use gradient based methods such as gradient descent back-propagation, Levenberg-Marquardt, Bayesian regularization back-propagation, and other choices which can be found in [34]. These algorithms are efficiently programmed in user friendly software packages such as MATLAB and PYTHON. □

4.5. Simulation Results

In order to implement VI algorithm for the ABS control, some preliminary actions are required to help improving the approximation quality of the neural networks. The first issue is the large ranges of variation of state variables in brake system model. For instance, considering slip ratio changing between 0 and 1, one can see that the vehicle traveled distance, vehicle velocity and braking pressure have totally different range of variations. This issue will result in poor training which should be resolved before initiation of the training process. One convenient solution for this problem is through nondimensionalization of the dynamics of the brake system. By nondimensionalization, the range of variations of each state variable falls into the interval $[0,1]$. Considering $x_z(.)$, $z \in \{1,2,3,4\}$, as

the state variables and $\max(x_z) = \mathscr{X}_z$, one can nondimensionalize $x_z(.)$ as $\bar{x}_z(.) = \frac{x_z(.)}{\mathscr{X}_z}$. Hence, the nondimensionalized dynamics from Eq. (4.9) can be represented as follows

$$\begin{aligned}
\dot{\bar{x}}_1(t) &= \frac{\mathscr{X}_2}{\mathscr{X}_1}\bar{x}_2(t) \\
\dot{\bar{x}}_2(t) &= \frac{1}{\mathscr{X}_2}\frac{-F_f(\bar{x}_3(t)\mathscr{X}_3)}{M} \\
\dot{\bar{x}}_3 &= \frac{1}{\mathscr{X}_2\mathscr{X}_3}\frac{-1}{\bar{x}_2(t)}[\frac{F_f(\bar{x}_3(t)\mathscr{X}_3)}{M}(1-\bar{x}_3(t)\mathscr{X}_3)+\frac{R^2F_f(\bar{x}_3(t)\mathscr{X}_3)}{I}]+\frac{\mathscr{X}_4}{\mathscr{X}_2\mathscr{X}_3}\frac{RK_b}{I\bar{x}_2(t)}\bar{x}_4(t) \\
\dot{\bar{x}}_4(t) &= \overline{m}_{vc}(\bar{x}(t)), vc \in \{1,2,3\}
\end{aligned} \tag{4.28}$$

where

$$\begin{cases}
\overline{m}_1(\bar{x}(t)) = \frac{1}{\mathscr{X}_4}\frac{A_1}{C_f}\sqrt{\frac{2}{\rho}(P_p-\bar{x}_4(t)\mathscr{X}_4)} \\
\overline{m}_2(\bar{x}(t)) = \frac{-1}{\mathscr{X}_4}\frac{A_2}{C_f}\sqrt{\frac{2}{\rho}(\bar{x}_4(t)\mathscr{X}_4-P_r)} \\
\overline{m}_3(\bar{x}(t)) = 0
\end{cases} \tag{4.29}$$

For simulation purpose, the maximum values for each state were selected as $\mathscr{X}_1 = 1000$ (m), $\mathscr{X}_2 = 100$ (m/s), $\mathscr{X}_3 = 1$ and $\mathscr{X}_4 = 107.15$ $(\mathbf{k}Pa)$.

For training neural networks, one needs to satisfy the so-called Persistency of Excitation (PE) condition in adaptive control literature [45]. One easy way to fulfill this goal is using random training patterns which would cover the domain of interest, Ω. In general, selection of proper random samples for training the neural networks plays a crucial role in the quality of training. Although the nondimensionalized state variables can change in $[0,1]$, application of small quantities for normalized vehicle velocity, $\bar{x}_2(.)$, leads to large or even infinite magnitudes for normalized $\dot{\lambda}(.)$ and $\lambda(.)$ accordingly, as discussed earlier in *Remark* 4.2.1. In order to prevent divergence in training due to improper choice of training samples, one can confine the region of training, Ω, such that the undesirable values for the state variables are not included. In this study, the region of training is confined as

$$\Omega = \{[\bar{x}_1,\bar{x}_2,\bar{x}_3,\bar{x}_4]|\quad 0 \le \bar{x}_1 \le 1,\ 0.1 \le \bar{x}_2 \le 0.9,\ 0 \le \bar{x}_3 \le 0.5,\ 0 \le \bar{x}_4 \le 0.56\} \tag{4.30}$$

Table 4.2: Parameters of ABS model for simulation

Parameter	**Magnitude**
M	500 $(\mathbf{k}g)$
I	1.6 $(\mathbf{k}g.m^2)$
R	0.3 (m)
K_b	28
A_1	0.0316 (m^2)
A_2	0.0633 (m^2)
ρ	850 $(\mathbf{k}g/m^3)$
P_p	6000 $(\mathbf{k}Pa)$
P_r	5 $(\mathbf{k}Pa)$
C_f	0.00158 $(m^{9/2}s/\mathbf{k}g)$
g	9.81 (m/s^2)

In order to start the simulations, one needs to define the parameters of ABS model. These parameters were chosen as in Table 4.2. It is also required to define the desired value for slip ratio. Considering Eq. (4.6), one can find the desired slip as the slip which maximizes the longitudinal force. Since $F_f(.)$ is a linear function of friction coefficient, the desired slip ratio is the one which maximizes $\mu(\lambda)$. Hence, through differentiating Eq. (4.6) with respect to λ one has

$$\frac{d\mu(\lambda)}{d\lambda} = \mu_H(C_1C_2e^{-C_2\lambda} - C_3) \tag{4.31}$$

Equating Eq. (4.31) to zero, one can define the desired slip ratio, λ_d as

$$\lambda_d = -\frac{1}{C_2}Ln\Big(\frac{C_3}{C_1C_2}\Big) \tag{4.32}$$

For dry asphalt road, $C_1 = 1.28$, $C_2 = 23.99$ and $C_3 = 0.52$ [84] which results in $\lambda_d = 0.17$. Since the control goal is designing a feedback controller for minimizing the stopping distance of the vehicle and enforcing the vehicle velocity to zero, one expects that the amount of slip on-the-fly be

close to the desired value, i.e. $\lambda_d = 0.17$.

4.5.1. Implementation of VI Algorithm with LIP Neural Networks for ABS Control

In order to show the effectiveness of VI algorithm for optimal scheduling in ABS, LIP neural networks with polynomial basis functions were selected. As mentioned before, the structure of LIP networks greatly facilitates the training process and enables using least squares method for offline training in batch mode. Meanwhile, LIP networks imposes relatively smaller computational burden, compared to MLP networks, in online evaluation of the value function. This feature of LIP networks makes them a great, and most of the time the best, choice for approximating the value function in VI algorithm. For the first simulation, a set of 700 training samples was generated within Ω introduced previously and $\delta t = 0.01\ (s)$ was selected as the sample time for discretizing the brake system dynamics. The discount factor was selected as $\gamma = 0.9$. Since the control goal is designing a feedback controller for minimizing the stopping distance of the vehicle and enforcing the vehicle velocity to zero, state penalizing matrix, Q, was selected as a diagonal matrix with $[1000, 1000, 0, 0]$ on the main diagonal and 0 elsewhere. In other words, the stopping distance and the vehicle velocity are penalized only. Based on the discussed theory, by online using the optimal scheduler one expects that the amount of slip be close to the desired value, i.e. $\lambda_d = 0.17$.

In order to start the simulations, basis functions of the LIP network were selected as polynomials with all possible combinations of the state variables up to the 4th degree without repetitions which led to 34 neurons. The initial weight vector was selected as $\widehat{W}_{LIP}^{0} = \mathbf{0}$ where $\mathbf{0}$ denotes a vector of all elements as zeros with a proper dimension. In each iteration of VI algorithm, least squares method was used to find $\widehat{W}_{LIP}^{i+1}$. The training process converged in 95 iterations of VI algorithm. The History of the critic weights during the training process is shown in Fig. 4.4. Once the training process finished, the converged critic was selected for value function approximation and feedback scheduling on-the-fly without retraining. For this purpose, the initial state was selected as $\bar{x}(0) = [0, 0.7, 0, 0]$ which corresponds to 70 (m/s) vehicle initial velocity. The application of ABS controller was terminated when vehicle velocity reached the amount of 10 (m/s) which corresponds to the lower bound in domain of interest for vehicle velocity. The history of the states and the switching schedule

are shown in Fig. 4.5 and Fig. 4.6, respectively.

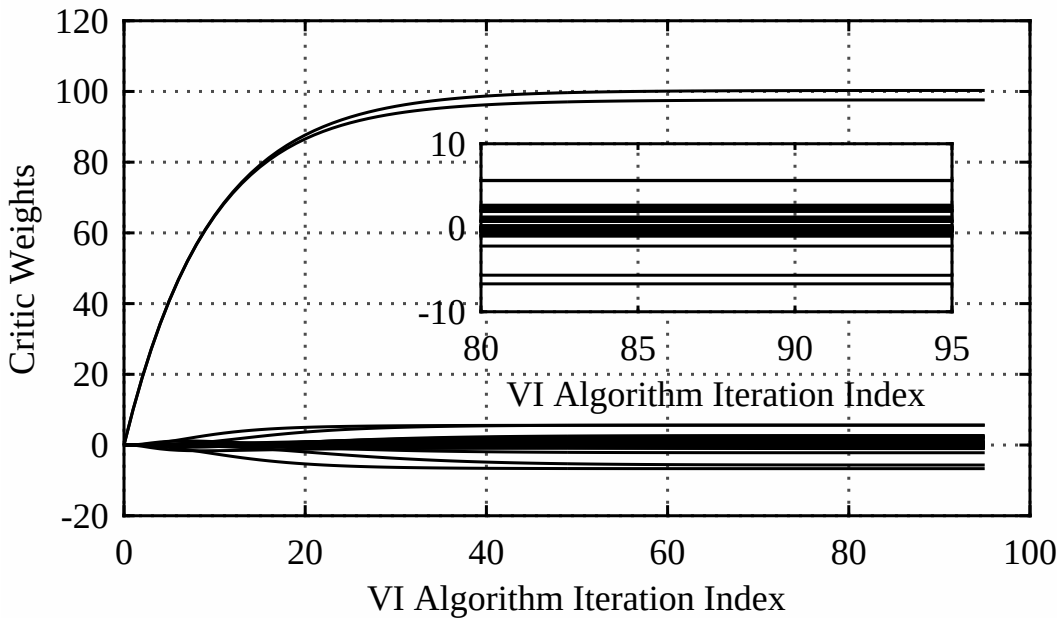

Figure 4.4: History of the LIP critic weights in offline training with least squares in batch mode.

As one can see from Fig. 4.5, the trained controller was able to stop the vehicle in 6.36 (s). The controller first activated mode 1 to increase the braking pressure and then tried to keep that pressure and avoid lock-ups by switching between mode 2 and 3, that is, pressure decrease and pressure hold, respectively. Using this controller, the vehicle stopping distance is 263.4 (m). Also, the slip ratio is tried to be regulated close to the desired value, i.e. $\lambda = 0.17$. The mean absolute error in desired slip ratio tracking is 0.0465 and the total number of switching is 146.

From theoretical point of view, one expects the approximation precision of LIP networks improves by increasing the number of neurons. In the case of VI algorithm, improvements in value function approximation leads to improvements in calculating the optimal solution and better performance of the trained controller. However, increasing the number of neurons after a certain number would start to have a negative effect on the quality of training with least squares in batch mode. This is mainly because of computational concerns in matrix inversion required in training with least squares in batch mode. For this reason, increasing the number of neurons more than 34 neurons did not show a significant improvement in control of ABS. In order to improve the

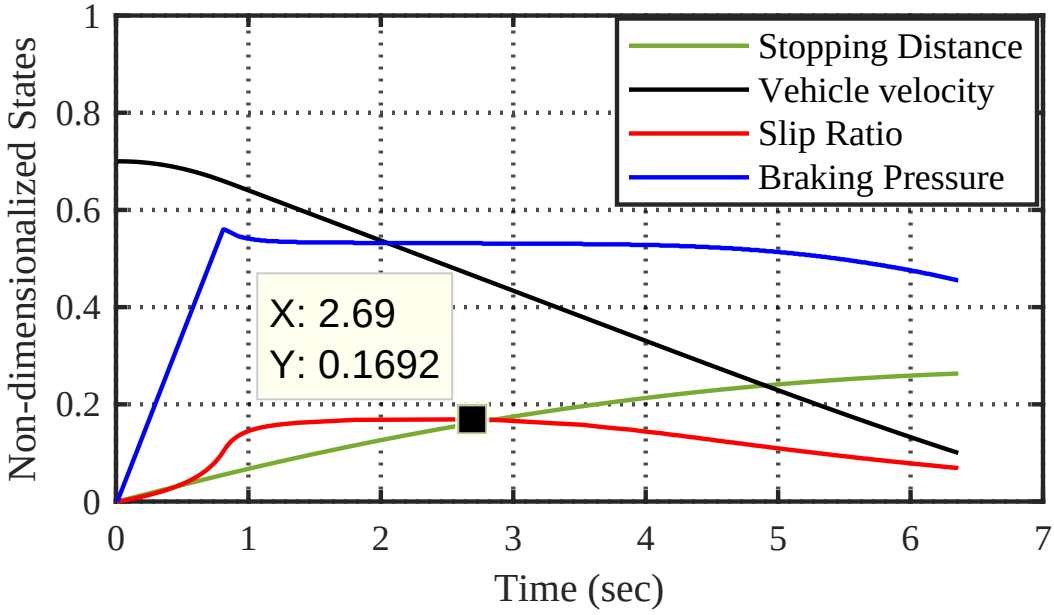

Figure 4.5: History of the states using the trained LIP neural network controller with 34 neurons and initial condition $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

approximation capability of function approximators in implementing VI algorithm, application of MLP neural networks is studied in the next section.

4.5.2. Implementation of VI Algorithm with MLP Neural Network for ABS Control

The structure of MLP neural networks was discussed in section 4.4. In order to improve the performance of the controller, MLP neural networks were used in VI algorithm to approximate the value function. For this purpose, a MLP network with three hidden layers was selected. The number of neurons in the first, second and third hidden layers were 15, 10, and 5, respectively. The activation functions in the hidden layers which preform the nonlinear mapping were selected as Gaussian functions[1]. For the output layer, activation function was chosen as a linear function[2]. For batch mode offline training, Bayesian regularization back-propagation algorithm was selected from MATLAB neural networks toolbox. The initial parameters of the MLP network including the weight and bias terms were selected randomly for the first iteration of VI algorithm.

[1] $\mathscr{F}(x) = \frac{2}{1+exp(-2x)} - 1$

[2] $\mathscr{G}(x) = x$

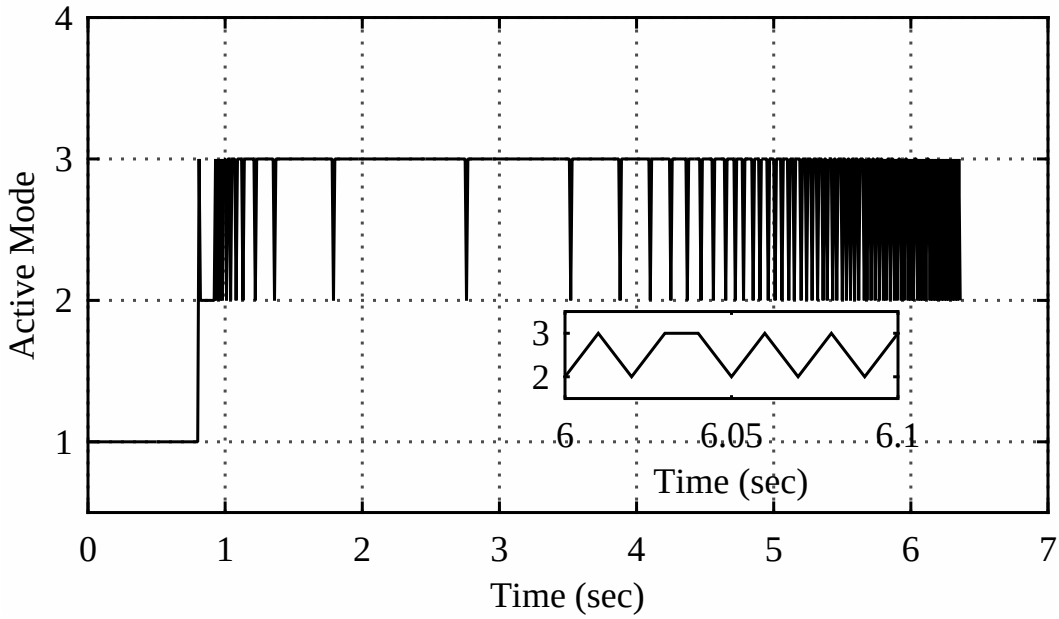

Figure 4.6: Switching schedule using the trained LIP neural network controller with 34 neurons and initial condition $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

In order to start the training process, a set of 1000 random samples was generated in the same domain of interest as used for the previous example. Meanwhile, for improving the tracking capability of the proposed controller, the state penalizing matrix was chosen such that only deviation of the slip ratio from the optimal value, calculated earlier to be 0.17, is penalized. Hence, Q was selected as a diagonal matrix with $[0, 0, 1000, 0]$ on the main diagonal and 0 elsewhere. Once the training process finished, converged critic was used for online control with the same initial condition used in the previous example. The state trajectories and switching schedule are shown in Fig. 4.7 and Fig. 4.8, respectively.

Comparing the performance of the MLP network in Fig. 4.7 with that of the LIP network in Fig. 4.5, one can easily see that the MLP controller showed significantly better performance in regulating λ about the desired value, $\lambda_d = 0.17$. As a summary for the performance of this controller the stopping time and distance were 6.219 (s) and 260.6 (m), respectively. Meanwhile, the mean absolute error for desired slip tracking was 0.0205. As one can see, improvement in value function approximation of VI algorithm led to better performance of the closed loop system.

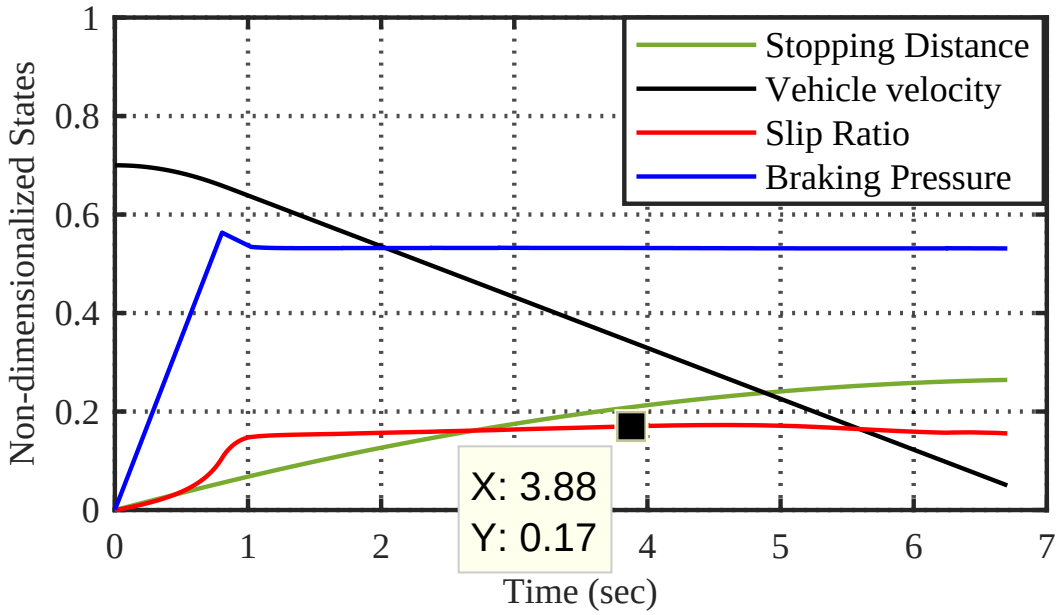

Figure 4.7: History of the states using the trained MLP neural network controller with 3 hidden layers, $[15,10,5]$ neurons and initial condition $\bar{x}(0) = [0,0.7,0,0]^T$.

Another interesting feature of the MLP controller is the fewer number of switching it requires for regulating the slip ratio. As a quick review, the number of switching in the LIP controller with 34 neurons was 146, and in MLP controller with 3 hidden layers was 80. High frequency switching is usually undesirable. In the general case of mechanical systems, one is interested to reduce the number of switching to avoid possible problems such as actuator saturation or bandwidth concerns. In case of ABS control, hydraulic brake systems operate with delays and they have distinctively different rates in oil filling and emptying dynamics. Considering these two characteristics of hydraulic brake systems, high frequency switching may not be possible to apply due to saturation. In order to decrease switching frequency, different remedies have been suggested [39]. In this study, minimum dwell time remedy is applied.

4.5.3. Remedying the High Frequency Switching Issue

In minimum dwell time remedy the system is allowed to switch if at least a certain amount of time, called dwell time and denoted by Δt, is elapsed from the previous switching. For instance, consider the system had a switching at time instant t_1 and at the present time instant, t, system needs

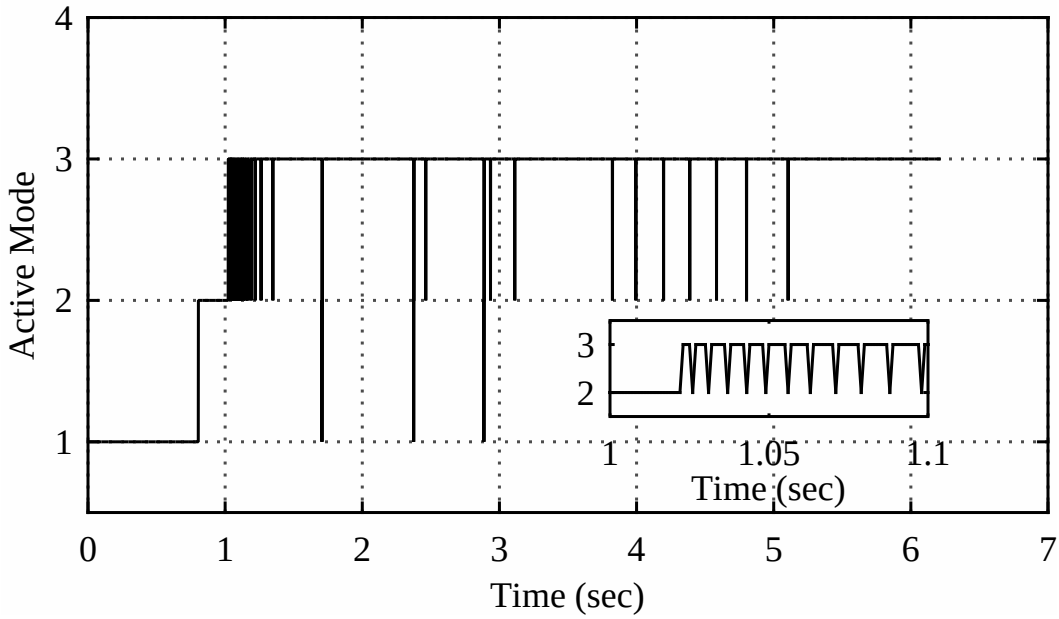

Figure 4.8: Switching schedule using the trained MLP neural network controller with 3 hidden layers, $[15,10,5]$ neurons and initial condition $\bar{x}(0) = [0,0.7,0,0]^T$.

to switch. Also, consider that minimum dwell time remedy is applied on the system. If $t - t_1 \geq \Delta t$, system is permitted to change its mode and switch. On the other hand, if $t - t_1 < \Delta t$ system cannot change its mode and it needs to continue to use the same mode until its dwell time is elapsed.

In this section, the performance of the MLP neural network controller with 3 hidden layers operating with minimum dwell time remedy is investigated. The minimum dwell time is chosen as $\Delta t = 0.03\ (s)$. In order to challenge the controller, and also guarantee the performance of the controller in real-time implementations, the dwell time is chosen to be bigger than $0.007\ (s)$ which was frequently reported in recent published studies [46, 47, 70, 84]. The history of the states and the switching schedule resulted from applying the minimum dwell time remedy are shown in Fig. 4.9 and Fig. 4.10, respectively.

As a summary, the mean absolute error of the desired slip tracking is 0.0321. The vehicle stopping time and distance are $6.226\ (s)$ and $260.8\ (m)$ and the total number of switching is 72. As one can see in Fig. 4.10, the switching schedule is sparsely and nicely distributed throughout the simulation time and no high frequency region can be detected. However, high frequency switching

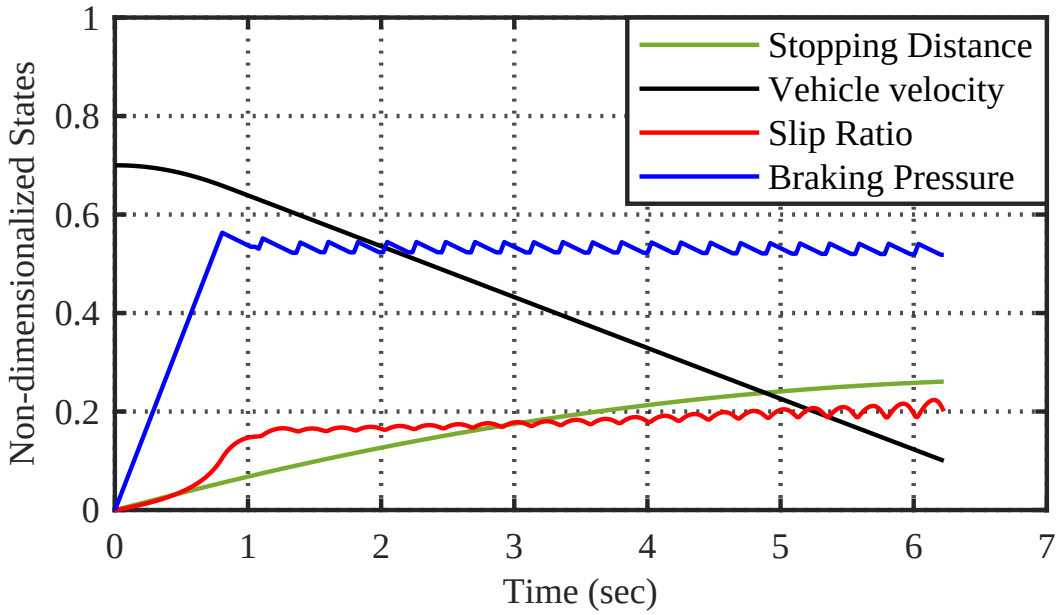

Figure 4.9: State trajectories resulted from using the trained MLP neural network controller with 3 hidden layers. The minimum dwell time was $\Delta t = 0.03\ (s)$, and initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

can be seen in Fig. 4.8, see the region at $1 < t < 1.5$, where the minimum dwell time remedy was not applied. As a result, although application of minimum dwell time remedy led to close number of switching in both Fig. 4.10 and Fig. 4.8, as given in Table 4.3, the main advantage of this remedy is preventing high frequency switching which ensures the effectiveness of the designed controller in practical applications. Meanwhile, the controller showed a significantly good performance in terms of desired slip ratio tracking and was able to stop the vehicle in a short distance and time. At the end of this section, the simulation results for the discussed controllers are summarized in Table 4.3 for better comparison.

4.5.4. Robustness Evaluation

As mentioned in section 4.2, for modeling the dynamics of the brake system steering effects and drag forces were neglected. Also, in order to derive a simplified model of the brake system the effect of load transfer in braking was neglected as discussed in Remark 4.2.3. In general, it is important to consider the effect of these modeling uncertainties on the performance of the designed

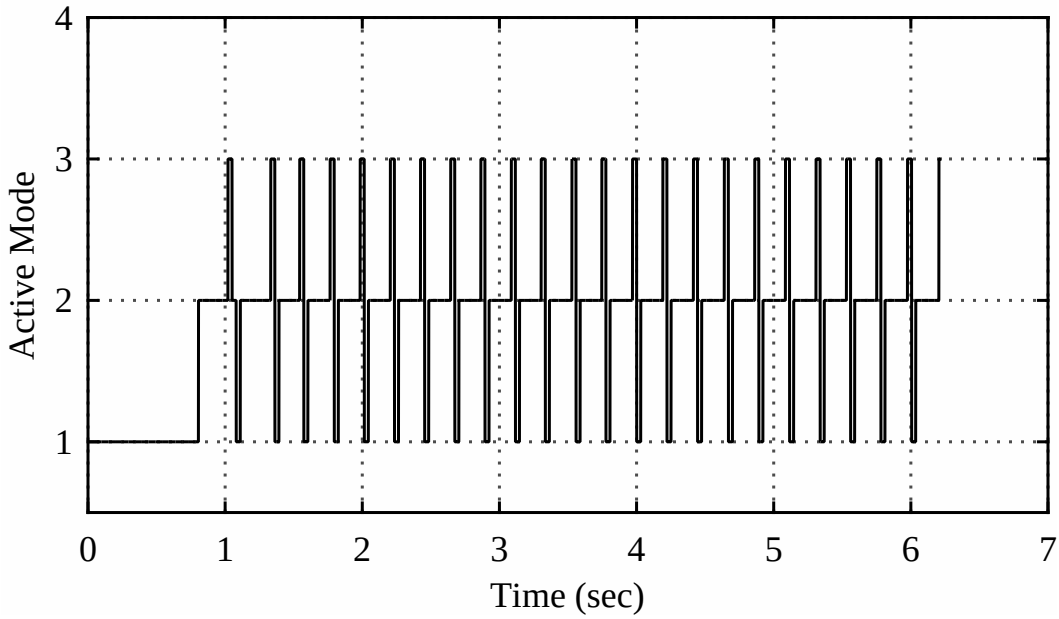

Figure 4.10: Switching schedule using the trained MLP neural network controller with 3 hidden layers. The minimum dwell time was $\Delta t = 0.03\ (s)$, and initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

Table 4.3: Summary of Simulation Results

	LIP ($m = 34$)	**MLP** ($m = [15, 10, 5]$)	**MLP** ($m = [15, 10, 5]$ & $\Delta t = 0.03$)
Stopping Time (s)	6.36	6.219	6.226
Stopping Distance (m)	263.4	260.6	260.8
No. of Switching	146	80	72
Mean Absolute Error	0.0405	0.0205	0.0321

scheduler. To study the effect of the neglected dynamics on the performance of the controller, one can consider uncertainty on the longitudinal force. Also, the effect of minimum dwell time remedy for preventing high frequency switching was discussed in section 4.5.3. As shown in section 4.5.3, minimum dwell time remedy affects the controller performance and it can be considered as an input disturbance/uncertainty. Hence, in this section the effect of uncertainties in modeling and control on the performance of the optimal scheduler is discussed. Before starting the robustness evaluation, it is worthy of attention that in all subsequent simulations the controller which was trained offline in section 4.5.2 is used for online control without retraining. Also, the initial condition for all

simulations is selected as $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

4.5.4.1. Modeling Uncertainty

For evaluating the robustness of the controller, a random noise signal with magnitude in the range of $[-0.1F_f(.), +0.1F_f(.)]$ was added to the longitudinal force depicted in Eq. (4.7). Using the optimal scheduler, the history of the states is illustrated in Fig. 4.11. As a summary of the performance of this simulation example, the stopping time and distance are 6.225 (*sec*) and 260.8 (m), respectively. Also, the mean absolute error for desired slip ratio tracking is 0.0248 and the controller switched for 2950 times. Compared to the case without uncertainty, the stopping time is increased for about 0.1% and the stopping distance is increased for about 0.08%. The number of switching is about 37 times the case without uncertainty and mean absolute tracking error for the desired λ tracking increased for almost 21%.

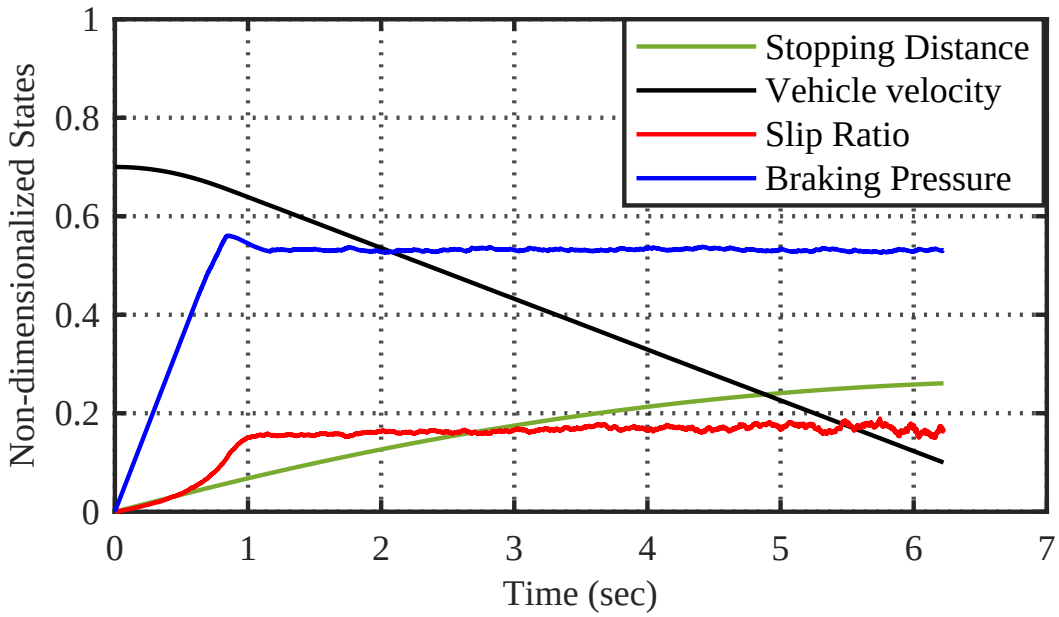

Figure 4.11: History of states using the trained MLP neural network controller with 3 hidden layers and the imposed uncertainty on the longitudinal force. The initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

In order to qualitatively verify whether the amount of the imposed uncertainty on the longitudinal force is enough to deteriorate the performance of the open loop system and also to verify how

effectively the closed loop system handles the uncertainty, the mode sequence depicted in Fig. 4.8 is given to the system while the uncertainty is imposed. For a better comparison, the same uncertainty was applied on the closed loop system as well. The performance of the open loop and closed loop system in terms desired slip ratio tracking is depicted in Fig. 4.12,[1]. As one can see from Fig. 4.12, wheel locked up in the open loop system after 4 (*sec*). However, the closed system kept the slip ratio close to the desired value. The history of the vehicle velocity is compared in Fig. 4.13 where the final values of the vehicle velocity for the open loop system is greater than the closed loop system. This means that the closed loop system was able to reduce the velocity more than the open loop system in the same duration of time.

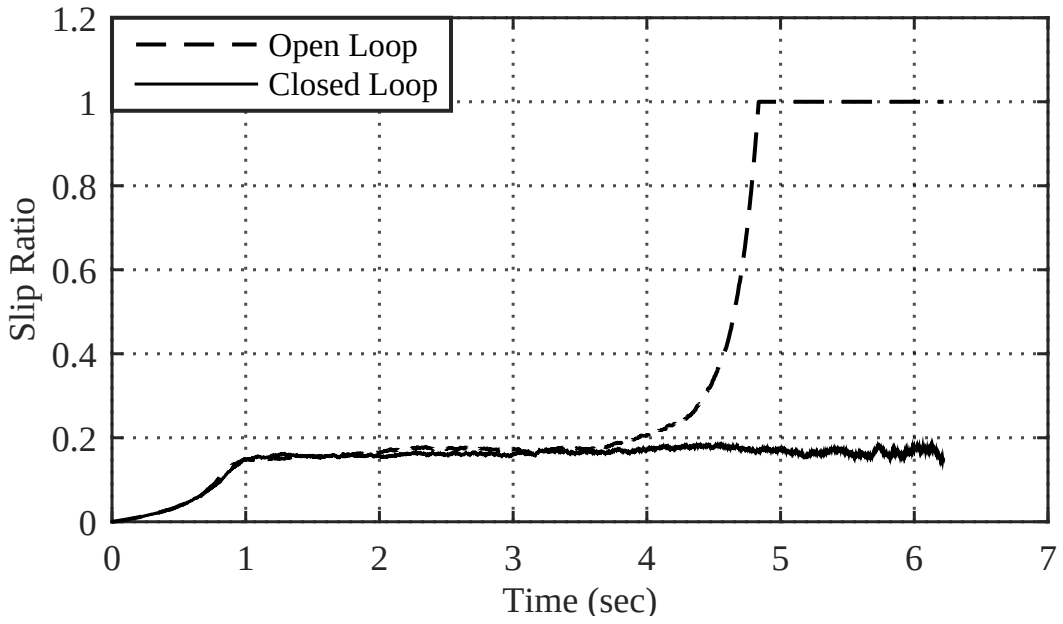

Figure 4.12: Comparison between the slip ratio generated by the open loop system and the closed loop system with the uncertainty imposed on the longitudinal force. The initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$. Wheel Lock-up occurred in the open loop system where the closed loop control was capable to keep the slip ration, i.e., λ, close to the desired value.

As one can see from the simulation results in this section, the controller tries to cancel the effect of imposed uncertainty by more switching. However, in switched systems high frequency switching

[1]Since the mode sequence in Fig. 4.8 is only for the simulation of 6.219 (*sec*), the simulations in this section are manually terminated after 6.219 (*sec*).

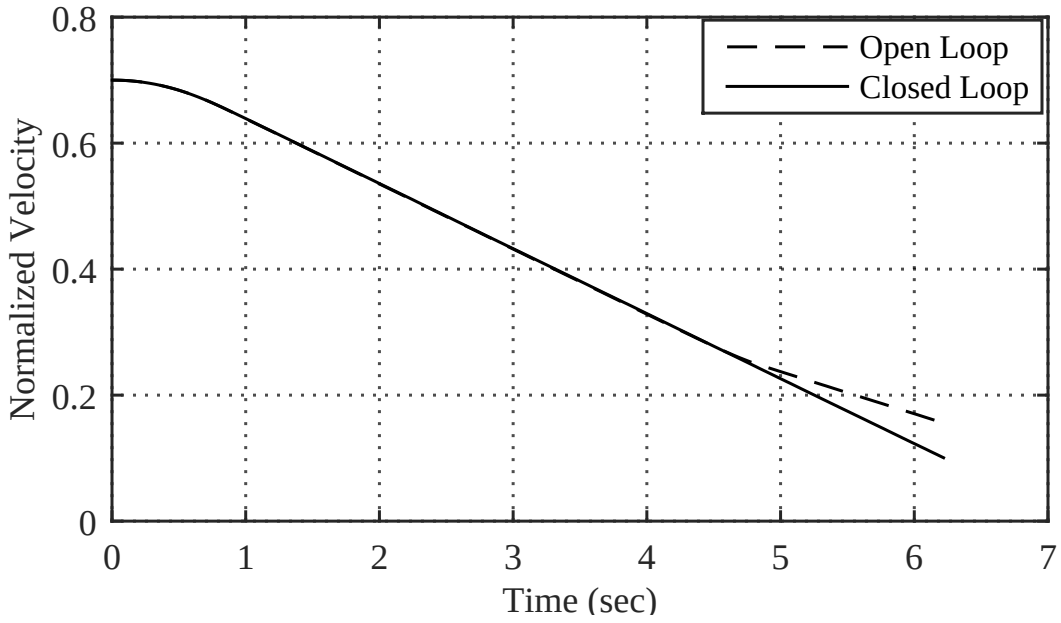

Figure 4.13: Comparison between the vehicle velocity in the open loop system and the closed loop system with the uncertainty imposed on the longitudinal force. The initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

is generally undesirable as discussed in section 4.5.3. Hence, in the next subsection the effect of minimum dwell time remedy in the presence of modeling uncertainty is investigated.

4.5.4.2. Modeling Uncertainty with Minimum Dwell Time Remedy

To investigate the effect of minimum dwell time remedy in the presence of modeling uncertainty, a noise signal with the magnitude in the range of $[-0.1F_f(.), +0.1F_f(.)]$ was added to the longitudinal force. The minimum dwell time was selected as $\Delta t = 0.007$ (sec). This minimum dwell time is frequently used in the literature [46, 47, 70, 84] for control of ABS. The controller which was trained in section 4.5.2 was used with the same initial condition selected for the previous simulations.

The history of the states is depicted in Fig. 4.14. As a summary of results, the stopping time and distance are 6.256 (sec) and 261.911 (m), respectively. The mean absolute error for desired slip ratio tracking is 0.0402 and the controller switched for 680 times. Compared to the case without uncertainty and dwell time, the stopping time is increased only for about 0.6% and the stopping distance is increased for about 0.5%. The number of switching is 8.5 times the case without

uncertainty and the mean absolute tracking error for the desired λ tracking is increased for about 96%.

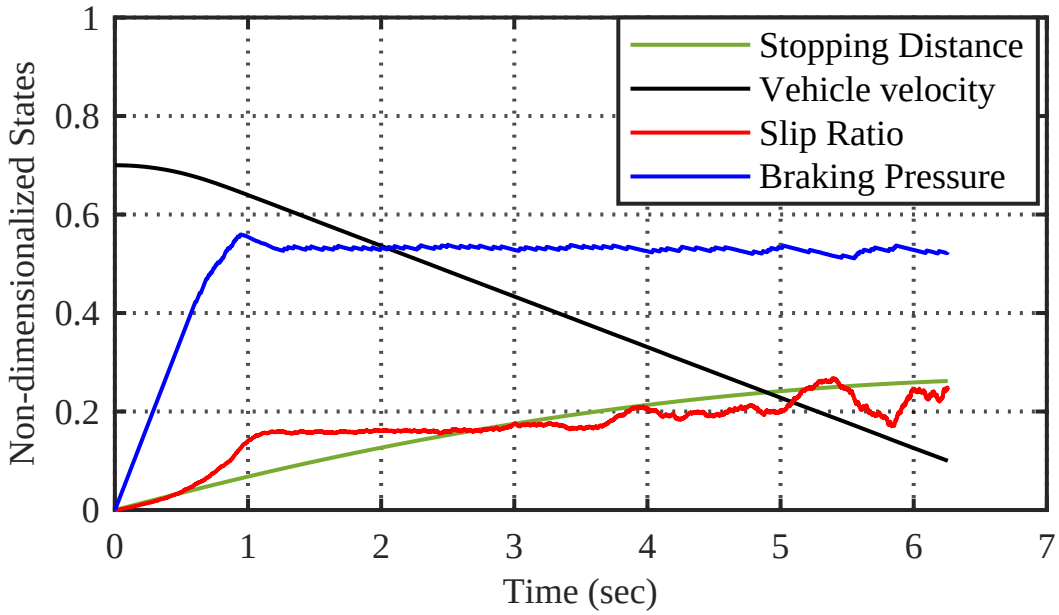

Figure 4.14: History of states using the trained MLP neural network controller with 3 hidden layers and the imposed uncertainty on the longitudinal force. The dwell time was $\Delta t = 0.007$ (sec) and initial condition was $\bar{x}(0) = [0, 0.7, 0, 0]^T$.

The robustness evaluation of the proposed controller is summarized in Table 4.4.

Table 4.4: Summary of Robustness Evaluations

	Modeling Uncertainty & $\Delta t = 0$	Modeling Uncertainty & $\Delta t = 0.007$
Stopping Time (s)	6.225	6.256
Stopping Distance (m)	260.8	261.911
No. of Switching	2950	680
Mean Absolute Error	0.0248	0.0402

Remark 4.5.1 It is of interest to compare the performance of the sub-optimal controller with the existing ones in the literature such as [46, 47, 70, 84]. Unfortunately, due to differences in modeling

the dynamics of ABS in this chapter and the presented works, this comparison is not possible. For example, one of the best control method to be compared with the one developed in this chapter is the on/off scheduler developed in [46]. In [46], the brake torque is modeled with constant rates of increasing and decreasing. This is in contrast with the model adapted in this chapter which considers varying, i.e. not constant, rate of change for braking torque. □

4.6. Conclusion

A feedback solution for optimal scheduling in anti-lock brake system of ground vehicles was developed based on approximate dynamic programming. The proposed solution used value iteration algorithm to learn the infinite horizon solution of the underlying Hamilton-Jacobi-Bellman equation. Meanwhile, the presented method explicitly incorporated the switching nature of anti-lock brake system. In order to investigate the effect of approximation errors in value function approximation stage of value iteration algorithm, two types of neural networks as linear in parameter and multi-layer neural networks were studied. It was shown that improvements in approximation precision in value function approximation leads to better performance of the trained controller in online control. Also, to ensure the effectiveness of the online scheduler in real-world implementations, the effect of minimum dwell time remedy for preventing the high frequency switching was studied.

4.7. Declaration on conflict of interest

The authors certify that there is no conflict of interest with any financial organization or personal interests regarding the material discussed in this study. This study was funded by National Science Foundation under Grant Number 1509778. The proposed study in the manuscript involved no experiment on any live existence such as humans or animals.

Chapter 5

Conclusion

Optimal switching in switched systems with autonomous subsystems and continuous-time dynamics was investigated. A policy iteration algorithm was introduced and the convergence to the optimal solution and stability of the system during the training was investigated. Three different training algorithms as offline training, online training, and concurrent training were introduced. For online training, two training laws based on gradient descent and recursive least squares algorithm were introduced. Also, a new PI algorithm was introduced which tries to reduce the computational burden in the PI algorithm. As another contribution, an algorithm for tracking a reference signal was introduced. At the end, an optimal switching scheduler for control of anti-lock brake systems in ground vehicles was introduced which uses a value iteration algorithm.

BIBLIOGRAPHY

[1] ABBOTT, S. *Understanding Analysis*. Springer New York, 2002.

[2] ABU-KHALAF, M., AND LEWIS, F. L. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica 41*, 5 (2005), 779 – 791.

[3] BANIA, P., KORYTOWSKI, A., SZYMKAT, M., AND GORCZYCA, P. Optimal control of a laboratory antilock brake system. *IFAC Proceedings Volumes 38*, 1 (2005), 290–295. 16th IFAC World Congress.

[4] BARDI, M., AND CAPUZZO-DOLCETTA, I. *Optimal control and viscosity solutions of Hamilton–Jacobi–Bellman equations*. Boston,MA, Birkhause, 1997.

[5] BEARD, R. W. *Improving the Closed-Loop Performance of Nonlinear Systems*. Ph.D. Dissertation at Rensselaer Polytechnic institute, 1995.

[6] BEARD, R. W., SARIDIS, G. N., AND WEN, J. T. Galerkin approximations of the generalized hamilton-jacobi-bellman equation. *Automatica 33*, 12 (1997), 2159 – 2177.

[7] BERTSEKAS, D. P. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems 28*, 3 (2017), 500–509.

[8] BERTSEKAS, D. P., AND TSITSIKLIS, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[9] BHASIN, S., KAMALAPURKAR, R., JOHNSON, M., VAMVOUDAKIS, K., LEWIS, F., AND DIXON, W. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica 49*, 1 (2013), 82 – 92.

[10] BIAN, T., AND JIANG, Z.-P. Value iteration and adaptive dynamic programming for data-driven adaptive optimal control design. *Automatica 71* (2016), 348 – 360.

[11] BORISOV, V. F. Fuller's phenomenon: Review. *Journal of Mathematical Sciences 100*, 4 (Jul 2000), 2311–2354.

[12] CANNARSA, P., AND FRANKOWSKA, H. Some characterizations of optimal trajectories in control theory. *SIAM Journal on Control and Optimization 29*, 6 (1991), 1322–1347.

[13] CHOI, S. B. Sensorless adaptive speed control of a permanent-magnet dc motor for anti-lock brake systems. *International Journal of Automotive Technology 12*, 2 (2011), 207–212.

[14] CHOWDHARY, G. *Concurrent learning for convergence in adaptive control without persistency of excitation*. PhD thesis, Georgia Institute of Technology, 2010.

[15] CHOWDHARY, G., YUCELEN, T., MÜHLEGG, M., AND JOHNSON, E. N. Concurrent learning adaptive control of linear systems with exponentially convergent bounds. *International Journal of Adaptive Control and Signal Processing 27*, 4 (2013), 280–301.

[16] CORNO, M., GERARD, M., VERHAEGEN, M., AND HOLWEG, E. Hybrid ABS control using force measurement. *IEEE Transactions on Control Systems Technology 20*, 5 (2012), 1223–1235.

[17] DADASHNIALEHI, A., BAB-HADIASHAR, A., CAO, Z., AND KAPOOR, A. Intelligent sensorless ABS for in-wheel electric vehicles. *IEEE Transactions on Industrial Electronics 61*, 4 (2014), 1957–1969.

[18] DEUR, J., PAVKOVIĆ, D., BURGIO, G., AND HROVAT, D. A model-based traction control strategy non-reliant on wheel slip information. *Vehicle System Dynamics 49*, 8 (2011), 1245–1265.

[19] DHARMATTI, S., AND RAMASWAMY, M. Hybrid control systems and viscosity solutions. *SIAM Journal on Control and Optimization 44*, 4 (2005), 1259–1288.

[20] DIERKS, T., AND JAGANNATHAN, S. Optimal control of affine nonlinear continuous-time systems. In *American Control Conference (ACC), 2010* (2010), pp. 1568 – 1573.

[21] DINÇMEN, E., GÜVENÇ, B. A., AND ACARMAN, T. Extremum-seeking control of ABS braking in road vehicles with lateral force improvement. *IEEE Transactions on Control Systems Technology 22*, 1 (2014), 230–237.

[22] DOUSTI, M., BASLAMISLI, S. C., ONDER, E. T., AND SOLMAZ, S. Design of a multiple-model switching controller for ABS braking dynamics. *Transactions of institute of measurement and control 37*, 5 (2014), 582–595.

[23] DRAKUNOV, S., OZGUNER, U., DIX, P., AND ASHRAFI, B. ABS control using optimum search via sliding modes. *IEEE Transactions on Control Systems Technology 3*, 1 (1995), 79–85.

[24] DUGOFF, H., FANCHER, P. S., AND SEGEL, L. *Tire performance characteristics affecting vehicle response to steering and braking control inputs*. Technical Report. Highway Safety Research Institute, Ann Arbor, Michigan, 1969.

[25] FORMENTIN, S., NOVARA, C., SAVARESI, S. M., AND MILANESE, M. Active braking control system design: The $D^2 - IBC$ approach. *IEEE/ASME Transactions on Mechatronics 20*, 4 (2015), 1573–1584.

[26] FRANKOWSKA, H. *Value function in optimal control*. ICTP Lecture Notes, Summer School of Mathematical Control Theory, 2001.

[27] FULLER, A. T. Relay control systems optimized for various performance criteria. In *Proc. IFAC Congress, Moscow* (1961), Butterworth, London, pp. 510–519.

[28] GOEBEL, R. Stabilizing a linear system with saturation through optimal control. *IEEE Transactions on Automatic Control 50*, 5 (May 2005), 650–655.

[29] GOLDAR, S. N., YAZDANI, M., AND SINAFAR, B. Concurrent learning based finite-time parameter estimation in adaptive control of uncertain switched nonlinear systems. *Journal of Control, Automation and Electrical Systems 28*, 4 (Aug 2017), 444–456.

[30] GONG, Z., LIU, C., FENG, E., WANG, L., AND YU, Y. Modelling and optimization for a switched system in microbial fed-batch culture. *Applied Mathematical Modelling 35*, 7 (2011), 3276 – 3284.

[31] GOSAVI, A. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Springer US, 2003.

[32] GUSTAFSSON, F. Automotive safety systems. *IEEE Signal Processing Magazine 26*, 4 (2009), 32–47.

[33] HARIFI, A., AGHAGOLZADEH, A., ALIZADEH, G., AND SADEGHI, M. Designing a sliding mode controller for slip control of antilock brake systems. *Transportation Research Part C: Emerging Technologies 16*, 6 (2008), 731–741.

[34] HAYKIN, S. *Neural Networks and Learning Machines*. Prentice Hall, 2009.

[35] HEYDARI, A. Convergence analysis of policy iteration. *arXiv:1505.05216* (2015).

[36] HEYDARI, A. Optimal scheduling for reference tracking or state regulation using reinforcement learning. *Journal of the Franklin Institute 352*, 8 (2015), 3285 – 3303.

[37] HEYDARI, A. Stability analysis of optimal adaptive control under value iteration using a stabilizing initial policy. *IEEE Transactions on Neural Networks and Learning Systems PP*, 99 (2018), 1–6.

[38] HEYDARI, A. Stability analysis of optimal adaptive control using value iteration with approximation errors. *IEEE Transactions on Automatic Control PP*, 99 (2018), 1–1.

[39] HEYDARI, A., AND BALAKRISHNAN, S. N. Optimal switching between autonomous subsystems. *Journal of the Franklin Institute 351* (2014).

[40] HEYDARI, A., AND BALAKRISHNAN, S. N. Optimal switching between autonomous subsystems. *Journal of the Franklin Institute 351*, 5 (2014), 2675 – 2690.

[41] HOÀNG, T. B., PASILLAS-LÉPINE, W., BERNARDINIS, A. D., AND NETTO, M. Extended braking stiffness estimation based on a switched observer, with an application to wheel-acceleration control. *IEEE Transactions on Control Systems Technology 22*, 6 (2014), 2384–2392.

[42] HORNIK, K., STINCHCOMBE, M., AND WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359 – 366.

[43] HOWLETT, P., PUDNEY, P., AND VU, X. Local energy minimization in optimal train control. *Automatica 45*, 11 (2009), 2692 – 2698.

[44] HUANG, C. K., AND SHIH, M. C. Design of a hydraulic anti-lock braking system (ABS) for a motorcycle. *Journal of Mechanical Science and Technology 24*, 5 (2010), 1141–1149.

[45] IOANNOU, P. A., AND SUN, J. *Robust Adaptive Control*. Prentice-Hall Inc, Upper Saddle River, NJ, USA, 1995.

[46] JING, H., LIU, Z., AND CHEN, H. A switched control strategy for antilock braking system with on/off valves. *IEEE Transactions on Vehicular Technology 60*, 4 (2011), 1470–1484.

[47] JOHANSEN, T. A., PETERSEN, I., KALKKUHL, J., AND LUDEMANN, J. Gain-scheduled wheel slip control in automotive brake systems. *IEEE Transactions on Control Systems Technology 11*, 6 (2003), 799–811.

[48] JOHANSSON, K. H., EGERSTEDT, M., LYGEROS, J., AND SASTRY, S. On the regularization of zeno hybrid automata. *Systems & Control Letters 38*, 3 (1999), 141 – 150.

[49] JOHANSSON, R., AND RANTZER, A., Eds. *Nonlinear and Hybrid Systems in Automotive Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[50] KAMALAPURKAR, R., ANDREWS, L., WALTERS, P., AND DIXON, W. E. Model-based reinforcement learning for infinite-horizon approximate optimal tracking. In *53rd IEEE Conference on Decision and Control* (Dec 2014), pp. 5083–5088.

[51] KAMALAPURKAR, R., REISH, B., CHOWDHARY, G., AND DIXON, W. E. Concurrent learning for parameter estimation using dynamic state-derivative estimators. *IEEE Transactions on Automatic Control 62*, 7 (July 2017), 3594–3601.

[52] KAMALAPURKAR, R., ROSENFELD, J. A., AND DIXON, W. E. Efficient model-based reinforcement learning for approximate online optimal control. *Automatica 74* (2016), 247 – 258.

[53] KAMALAPURKAR, R., WALTERS, P., AND DIXON, W. Concurrent learning-based approximate optimal regulation. In *52nd IEEE Conference on Decision and Control* (Dec 2013), pp. 6256–6261.

[54] KAMALAPURKAR, R., WALTERS, P., AND DIXON, W. E. Model-based reinforcement learning for approximate optimal regulation. *Automatica 64* (2016), 94 – 104.

[55] KAMGARPOUR, M., AND TOMLIN, C. On optimal control of non-autonomous switched systems with a fixed mode sequence. *Automatica 48*, 6 (2012), 1177–1181.

[56] KHALIL, H. *Nonlinear Systems*. Prentice-Hall, 2002.

[57] KIENCKE, U., AND NIELSEN, L. *Automotive Control Systems: For Engine, Driveline, and Vehicle*. Springer, 2005.

[58] KIRK, D. *Optimal Control Theory: An Introduction*. Dover Publications, 2004.

[59] LAMPERSKI, A., AND AMES, A. D. Lyapunov theory for zeno stability. *IEEE Transactions on Automatic Control 58*, 1 (2013), 100–112.

[60] ŁAWRYŃCZUK, M. Nonlinear predictive control of dynamic systems represented by Wiener–Hammerstein models. *Nonlinear Dynamics* (2016), 1–22.

[61] LEE, K., AND PARK, K. Optimal robust control of a contactless brake system using an eddy current. *Mechatronics 9*, 6 (1999), 615–631.

[62] LEWIS, F. L., AND VRABIE, D. Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE 9*, 3 (2009), 32–50.

[63] LI, K., CAO, J., AND YU, F. Nonlinear tire-road friction control based on tire model parameter identification. *International Journal of Automotive Technology 13*, 7 (2012), 1077–1088.

[64] LIBERZON, D. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, NJ, USA, 2011.

[65] LIN, C. M., AND HSU, C. F. Self-learning fuzzy sliding-mode control for antilock braking systems. *IEEE Transactions on Control Systems Technology 11*, 2 (2003), 273–278.

[66] LIN, H., AND ANTSAKLIS, P. J. Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Transactions on Automatic Control 54*, 2 (2009), 308–322.

[67] LIU, D., YANG, X., AND LI, H. Adaptive optimal control for a class of continuous-time affine nonlinear systems with unknown internal dynamics. *Neural Computing and Applications 23*, 7-8 (2013), 1843 – 1850.

[68] LU, W., ZHU, P., AND FERRARI, S. A hybrid-adaptive dynamic programming approach for the model-free control of nonlinear switched systems. *IEEE Transactions on Automatic Control PP*, 99 (2015), 1–1.

[69] LU, W., ZHU, P., AND FERRARI, S. A hybrid-adaptive dynamic programming approach for the model-free control of nonlinear switched systems. *IEEE Transactions on Automatic Control 61*, 10 (2016), 3203–3208.

[70] MAUER, G. F. A fuzzy logic controller for an ABS braking system. *IEEE Transactions on Fuzzy Systems 3*, 4 (1995), 381–388.

[71] MIRZAEI, A., MOALLEM, M., DEHKORDI, B. M., AND FAHIMI, B. Design of an optimal fuzzy controller for antilock braking systems. *IEEE Transactions on Vehicular Technology 55*, 6 (2006), 1725–1730.

[72] MIRZAEI, M., AND MIRZAEINEJAD, H. Optimal design of a non-linear controller for anti-lock braking system. *Transportation Research Part C: Emerging Technologies 24* (2012), 19–35.

[73] MIRZAEINEJAD, H., AND MIRZAEI, M. A novel method for non-linear control of wheel slip in anti-lock braking systems. *Control Engineering Practice 18*, 8 (2010), 918–926.

[74] MIRZAEINEJAD, H., AND MIRZAEI, M. Optimization of nonlinear control strategy for anti-lock braking system with improvement of vehicle directional stability on split-μ roads. *Transportation Research Part C: Emerging Technologies 46* (2014), 1–15.

[75] MISHRA, A., LANGBORT, C., AND DULLERUD, G. E. Team optimal control of stochastically switched systems with local parameter knowledge. *IEEE Transactions on Automatic Control 60*, 8 (2015), 2086–2101.

[76] MODARES, H., LEWIS, F., AND NAGHIBI-SISTANI, M. B. Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Transactions on Neural Networks and Learning Systems 24*, 10 (2013), 1513 – 1525.

[77] MODARES, H., LEWIS, F. L., AND NAGHIBI-SISTANI, M. B. Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. *IEEE Transactions on Neural Networks and Learning Systems 24*, 10 (Oct 2013), 1513–1525.

[78] MODARES, H., NAGHIBI-SISTANI, M. B., AND LEWIS, F. L. A policy iteration approach to online optimal control of continuous-time constrained-input systems. *ISA Transactions 52*, 5 (2013), 611 – 621.

[79] NG, C. T., SONG, D. P., AND CHENG, T. C. E. Optimal policy for inventory transfer between two depots with backlogging. *IEEE Transactions on Automatic Control 57*, 12 (2012), 3247–3252.

[80] NIKSHI, W. M., HOOVER, R., AND BEDILLION, M. Nonlinear control synthesis for parking control of mixed conventional/braking actuation mobile robots. In *2017 American Control Conference (ACC)* (May 2017), pp. 616–622.

[81] OKYAY, A., CIGEROGLU, E., AND BAŞLAMIŞLI, S. A. A new sliding-mode controller design methodology with derivative switching function for anti-lock brake system. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 227*, 11 (2013), 2487–2503.

[82] OLSON, B. J., SHAW, S. W., AND STÉPÁN, G. Stability and bifurcation of longitudinal vehicle braking. *Nonlinear Dynamics 40*, 4 (Jun 2005), 339–365.

[83] PAKNIYAT, A., AND CAINES, P. E. On the relation between the minimum principle and dynamic programming for classical and hybrid control systems. *IEEE Transactions on Automatic Control 62*, 9 (2017), 4347–4362.

[84] PETERSEN, I. *Wheel slip control in ABS brakes using gain scheduled optimal control with constraints*. PhD thesis, Norwegian university of science and technology, 2003.

[85] POURSAMAD, A. Adaptive feedback linearization control of antilock braking systems using neural networks. *Mechatronics 19*, 5 (2009), 767–773.

[86] POWELL, W. B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley, 2011.

[87] PROKHOROV, D., AND WUNSCH, D. Adaptive critic designs. *IEEE Transactions on Neural Networks 8* (1997), 997 – 1007.

[88] QIN, C., ZHANG, H., AND LUO, Y. Optimal tracking control of a class of nonlinear discrete-time switched systems using adaptive dynamic programming. *Neural Computing and Applications 24*, 3 (2014), 531–538.

[89] RADAC, M.-B., AND PRECUP, R.-E. Data-driven model-free slip control of anti-lock braking systems using reinforcement q-learning. *Neurocomputing In press* (2017).

[90] RAJAMANI, R. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer, 2011.

[91] RINEHART, M., DAHLEH, M., AND KOLMANOVSKY, I. Value iteration for (switched) homogeneous systems. *IEEE Transactions on Automatic Control 54*, 6 (2009), 1290 – 1294.

[92] RINEHART, M., DAHLEH, M., REED, D., AND KOLMANOVSKY, I. Suboptimal control of switched systems with an application to the DISC engine. *IEEE Transactions on Control Systems Technology 16*, 2 (2008), 189–201.

[93] RUDIN, W. *Principles of Mathematical Analysis*, 3rd ed. McGraw-Hill, 1976.

[94] SARDARMEHNI, T., AND HEYDARI, A. Optimal switching in anti-lock brake systems of ground vehicles based on approximate dynamic programming. In *ASME 2015 Dynamic Systems and Control Conference (DSCC 2015)* (2015), pp. V003T50A010–V003T50A020.

[95] SARDARMEHNI, T., AND HEYDARI, A. Approximate solution for optimal control of continuous-time switched systems. In *ASME 2016 Dynamic Systems and Control Conference (DSCC 2016)* (2016), pp. V001T02A004–V001T02A014.

[96] SARDARMEHNI, T., AND HEYDARI, A. Policy iteration for optimal switching with continuous-time dynamics. In *Neural Networks (IJCNN), 2016 International Joint Conference on* (2016), IEEE, pp. 3536–3543.

[97] SARDARMEHNI, T., AND HEYDARI, A. Policy iteration for optimal switching with continuous-time dynamics. In *2016 International Joint conference on Neural Networks (IJCNN 2016)* (2016), pp. 3536–3543.

[98] SARDARMEHNI, T., AND HEYDARI, A. Suboptimal scheduling in switched systems with continuous-time dynamics: A least squares approach. *IEEE Transactions on Neural Networks and Learning Systems PP*, 99 (2017), 1–12.

[99] SARDARMEHNI, T., RAHMANI, H., AND MENHAJ, M. B. Robust control of wheel slip in anti-lock brake system of automobiles. *Nonlinear Dynamics 76*, 1 (2014), 125–138.

[100] SHAIKH, M. S., AND CAINES, P. E. On the hybrid optimal control problem: Theory and algorithms. *IEEE Transactions on Automatic Control 52*, 9 (2007), 1587–1603.

[101] SOKOLOV, Y., KOZMA, R., WERBOS, L. D., AND WERBOS, P. J. Complete stability analysis of a heuristic approximate dynamic programming control design. *Automatica 59* (2015), 9 – 18.

[102] STRULOVICI, B., AND SZYDLOWSKI, M. On the smoothness of value functions and the existence of optimal strategies in diffusion models. *Journal of Economic Theory 159, Part B* (2015), 1016 – 1055. Symposium Issue on Dynamic Contracts and Mechanism Design.

[103] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[104] SUZUKI, K. Optimal switching strategy of a mean-reverting asset over multiple regimes. *Automatica 67* (2016), 33 – 45.

[105] SVENDENIUS, J. *Tire modeling and friction estimation*. PhD thesis, Lund University, 2007.

[106] TANELLI, M., ASTOLFI, A., AND SAVARESI, S. M. Robust nonlinear output feedback control for brake by wire control systems. *Automatica 44*, 4 (2008), 1078–1087.

[107] TOPALOV, A. V., ONIZ, Y., KAYACAN, E., AND KAYNAK, O. Neuro-fuzzy control of antilock braking system using sliding mode incremental learning algorithm. *Neurocomputing 74*, 11 (2011), 1883–1893.

[108] TYUKIN, I., PROKHOROV, D., AND VAN LEEUWEN, C. A new method for adaptive brake control. In *American Control Conference.* (2005), vol. 3, pp. 2194–2199.

[109] VAMVOUDAKIS, K. G., AND HESPANHA, J. P. Online optimal switching of single phase DC/AC inverters using partial information. In *2014 American Control Conference (ACC)* (2014), pp. 2624 – 2630.

[110] VAMVOUDAKIS, K. G., AND LEWIS, F. L. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica 46*, 5 (2010), 878 – 888.

[111] VAMVOUDAKIS, K. G., VRABIE, D., AND LEWIS, F. L. Online adaptive algorithm for optimal control with integral reinforcement learning. *International Journal of Robust and Nonlinear Control 24*, 17 (2014), 2686 – 2710.

[112] VAN DER SCHAFT, A. J. L2-gain analysis of nonlinear systems and nonlinear state-feedback h_∞ control. *IEEE Transactions on Automatic Control 37*, 6 (Jun 1992), 770–784.

[113] VRABIE, D., AND LEWIS, F. L. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks 22*, 3 (2009), 237 – 246.

[114] VRABIE, D., PASTRAVANU, O., ABU-KHALAF, M., AND LEWIS, F. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica 45*, 2 (2009), 477 – 484.

[115] WANG, B., HUANG, X., WANG, J., GUO, X., AND ZHU, X. A robust wheel slip ratio control design combining hydraulic and regenerative braking systems for in-wheel-motors-driven electric vehicles. *Journal of the Franklin Institute 352*, 2 (2015), 577–602. Special Issue on Control and Estimation of Electrified vehicles.

[116] WANG, F.-Y., ZHANG, H., AND LIU, D. Adaptive dynamic programming: An introduction. *IEEE computational intelligence magazine 4*, 2 (2009).

[117] WEI, Q., LIU, D., AND LIN, H. Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. *IEEE Transactions on Cybernetics 46*, 3 (2016), 840–853.

[118] WERBOS, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University Press, 1974.

[119] WERBOS, P. J. Approximate dynamic programming for real-time control and neural modeling. In *Handbook of Intelligent Control*, D. A. White and D. A. Sofge, Eds. Multiscience Press, 1992.

[120] XIAO, G., ZHANG, H., AND LUO, Y. Online optimal control of unknown discrete-time nonlinear systems by using time-based adaptive dynamic programming. *Neurocomputing 165* (2015), 163 – 170.

[121] XIAO, G., ZHANG, H., AND LUO, Y. Online optimal control of unknown discrete-time nonlinear systems by using time-based adaptive dynamic programming. *Neurocomputing 165* (2015), 163 – 170.

[122] XIAO, G., ZHANG, H., LUO, Y., AND JIANG, H. Data-driven optimal tracking control for a class of affine non-linear continuous-time systems with completely unknown dynamics. *IET Control Theory Applications 10*, 6 (2016), 700–710.

[123] XIAO, G., ZHANG, H., LUO, Y., AND QU, Q. General value iteration based reinforcement learning for solving optimal tracking control problem of continuous–time affine nonlinear systems. *Neurocomputing 245* (2017), 114 – 123.

[124] XIAO, G., ZHANG, H., LUO, Y., AND QU, Q. General value iteration based reinforcement learning for solving optimal tracking control problem of continuous–time affine nonlinear systems. *Neurocomputing 245* (2017), 114 – 123.

[125] XU, X., AND ANTSAKLIS, P. J. Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. *International Journal of Control 75*, 16-17 (2002), 1406–1426.

[126] XU, X., AND ANTSAKLIS, P. J. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Transactions on Automatic Control 49*, 1 (2004), 2–16.

[127] YU, C., AND SHIM, T. Modeling of comprehensive electric drive system for a study of regenerative brake system. In *2013 American Control Conference* (2013), pp. 6734–6739.

[128] ZHANG, H., CUI, L., ZHANG, X., AND LUO, Y. Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Transactions on Neural Networks 22*, 12 (2011), 2226–2236.

[129] ZHANG, H., JIANG, H., LUO, C., AND XIAO, G. Discrete-time nonzero-sum games for multiplayer using policy-iteration-based adaptive dynamic programming algorithms. *IEEE Transactions on Cybernetics PP*, 99 (2017), 1–10.

[130] ZHANG, H., LUO, Y., AND LIU, D. Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Transactions on Neural Networks 20*, 9 (2009), 1490–1503.

[131] ZHANG, H., WEI, Q., AND LUO, Y. A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 38*, 4 (2008), 937–942.

[132] ZHANG, J., JOHANSSON, K. H., LYGEROS, J., AND SASTRY, S. Zeno hybrid systems. *International Journal of Robust and Nonlinear Control 11*, 5 (2001), 435–451.

[133] ZHANG, J., LV, C., GOU, J., AND KONG, D. Cooperative control of regenerative braking and hydraulic braking of an electrified passenger car. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 226*, 10 (2012), 1289–1302.

[134] ZHANG, W., HU, J., AND ABATE, A. On the value functions of the discrete-time switched lqr problem. *IEEE Transactions on Automatic Control 54*, 11 (2009), 2669–2674.

[135] ZHAO, D., WANG, B., AND LIU, D. A supervised actor–critic approach for adaptive cruise control. *Soft Computing 17*, 11 (Nov 2013), 2089–2099.

[136] ZHU, F., AND ANTSAKLIS, P. J. Optimal control of hybrid switched systems: A brief survey. *Discrete Event Dynamic Systems 25*, 3 (2014), 345–364.